Electrónica

Diseño Lógico

Fundamentos en electrónica digital

Flórez

ediciones de la U
conocimiento a su alcance

Electrónica

Diseño lógico

Fundamentos en electrónica digital

Héctor Arturo **Flórez** Fernández

ediciones de la U
conocimiento a su alcance
www.edicionesdelau.com

Flórez Fernández, Héctor Arturo
Diseño lógico: fundamentos de electrónica digital / Héctor Arturo Flórez Fernández. -- Bogotá : Ediciones de la U, 2010.
166 p. ; 24 cm.
ISBN 978-958-994-900-9
1. Electrónica digital 2. Dispositivos de almacenamiento (Computadores) 3. Algebra booleana 4. Computadores electrónicos digitales - Diseño y construcción I. Tít.
621.39 cd 21 ed.
A1253730

CEP-Banco de la República-Biblioteca Luis Ángel Arango

Área: Electrónica
Primera edición: Bogotá, Colombia, julio de 2010
ISBN. 978-958-994-900-9

© Héctor Arturo Flórez Fernández, www.hectorflorez.com
(Foros de discusión, blog del libro y materiales complementarios del autor en www.edicionesdelau.com)

© Ediciones de la U - Calle 24 A No. 43-22 - Tel. (+57-1) 4810505, Ext. 114
www.edicionesdelau.com - E-mail: editor@edicionesdelau.com
Bogotá, Colombia

Ediciones de la U es una empresa editorial que, con una visión moderna y estratégica de las tecnologías, desarrolla, promueve, distribuye y comercializa contenidos, herramientas de formación, libros técnicos y profesionales, e-books, e-learning o aprendizaje en línea, realizados por autores con amplia experiencia en las diferentes áreas profesionales e investigativas, para brindar a nuestros usuarios soluciones útiles y prácticas que contribuyan al dominio de sus campos de trabajo y a su mejor desempeño en un mundo global, cambiante y cada vez más competitivo.

Coordinación editorial: Adriana Gutiérrez M.
Carátula: Hipertexto Ltda.
Impresión:X-Press Estudio Gráfico Digital
Avda. Américas No. 39-53, Pbx. 6020808, Bogotá

Impreso y hecho en Colombia
Printed and made in Colombia

No está permitida la reproducción total o parcial de este libro, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro y otros medios, sin el permiso previo y por escrito de los titulares del Copyright.

Contenido

INTRODUCCIÓN 13
GLOSARIO 15

Capítulo 1. Sistemas Numéricos 19
1.1 SISTEMA DECIMAL 19
1.2 SISTEMA BINARIO 20
1.3 SISTEMA OCTAL 21
1.4 SISTEMA HEXADECIMAL 21
1.5 CÓDIGO BCD 22
1.6 CÓDIGO GRAY 23
1.7 CONVERSIÓN ENTRE SISTEMAS 24
1.7.1 Conversión Decimal-Binario 25
1.7.2 Conversión Binario-Decimal 26
1.7.3 Conversión Decimal-Octal 26
1.7.4 Conversión Octal-Decimal 28
1.7.5 Conversión Decimal-Hexadecimal 28
1.7.6 Conversión Hexadecimal-Decimal 30
1.7.7 Conversión Binario-Octal 30
1.7.8 Conversión Octal-Binario 31
1.7.9 Conversión Binario-Hexadecimal 31
1.7.10 Conversión Hexadecimal-Binario 32
1.8 COMPLEMENTO A 1 Y COMPLEMENTO A 2 32
1.8.1 Complemento a 1 33
1.8.2. Complemento a 2 33
1.9 OPERACIONES ARITMÉTICAS DE DIFERENTES SISTEMAS 34
1.9.1 Suma en Binario 34
1.9.2 Suma en Octal 34
1.9. 3 Suma en hexadecimal 35
1.9.4 Resta en Binario 36
1.9.5 Resta en octal 37
1.9.6 Resta en hexadecimal 38
1.9.7 Multiplicación en binario 38
1.9.8 División en binario 41
Ejercicios propuestos 44

Capítulo 2. Compuertas lógicas 45
2.1 COMPUERTA NOT 45
2.2 COMPUERTA AND 47
2.3 COMPUERTA OR 49
2.4. COMPUERTA NAND 51
2.5 COMPUERTA NOR 54
2.6 COMPUERTA OR EXCLUSIVA 56
2.7 COMPUERTA NOR EXCLUSIVA 58
2.8. IMPLEMENTACIóN MEDIANTE COMPUERTAS LóGICAS 61
Ejercicios Propuestos 64

Capítulo 3. Álgebra de Boole 65
3.1 OPERACIONES BOOLEANAS 65
3.1.1 Adición Booleana 65
3.1.2 Multiplicación Booleana 66
3.2 LEYES DEL ÁLGEBRA DE BOOLE 66
3.2.1 Ley Conmutativa 66
3.2.2 Ley Asociativa 67
3.2.3 Ley Distributiva 67
3.3 REGLAS DEL ÁLGEBRA DE BOOLE 68
3.4 TEOREMAS DE DeMORGAN 70
3.4.1 Aplicación de los teoremas de DeMorgan 71
3.5 SIMPLIFICACIÓN MEDIANTE EL ÁLGEBRA DE BOOLE 71
3.6 MAPAS DE KARNAUGH 72
3.6.1 Mapa de Karnaugh de tres variables 73
3.6.2 Mapa de Karnaugh de cuatro variables 73
3.6.3 Minimización de suma de productos mediante un mapa de Karnaugh 74
3.6.4 Simplificación de suma de productos mediante el mapa de Karnaugh 75
3.6.5 Simplicación de suma de productos usando Tabla de Verdad 77
3.6.6 Condiciones indiferentes o valores No importa 78
Ejercicios Propuestos 80

Capítulo 4. Lógica combinacional 83
4.1 SUMADOR BÁSICO 83
4.2 SUMADOR COMPLETO 84
4.3 SUMADOR BINARIO EN PARALELO 85
4.4 RESTADOR DE 4 BITS 86
4.5 COMPARADORES 88
4.6 CONVERSORES DE CÓDIGO 91
4.6.1 Conversor Binario-Gray 91
4.6.2 Conversor Binario-BCD 92
4.7 DECODIFICADORES 95
4.7.1 Decodificador 2-4 95
4.7.2 Decodificador 3-8 96

4.7.3 Decodificador Manejador 98
4.8 CODIFICADORES 101
4.8.1 Codificador Decimal BCD 101
4.8.2 Codificador Decimal BCD con prioridad 101
4.9 DEMULTIPLEXORES 103
4.10 MULTIPLEXORES 103
4.10.1 Multiplexor 4-1 103
4.10.2 Multiplexor cuádruple 2-1 105
Ejercicios Propuestos 107

Capítulo 5. Elementos básicos de almacenamiento 109
5.1 LATCHES 109
5.1.1 Latch S-R 109
5.1.2 Latch $\overline{S} - \overline{R}$ 111
5.1.3 Circuito Antirrebote 113
5.1.4 Latch S-R con Habilitación 114
5.1.5 Latch D con Habilitación 115
5.2 FLIP-FLOPS 116
5.2.1 Detector de flancos 116
5.2.2 Flip-Flop D 117
5.2.3 Flip-Flop J-K 117
5.2.4 Flip-Flop J-K con entradas asíncronas 118
5.3 TEMPORIZADOR 555 119
5.3.1 Temporizador 555 configurado como aestable 120
5.4 DIVISOR DE FRECUENCIA 122
Ejercicios Propuestos 124

Capítulo 6. Contadores 125
6.1 CONTADOR ASÍNCRONO 125
6.1.1 Contador asíncrono binario de 2 bits 125
6.1.2 Contador Asíncrono Binario de 4 bits 126
6.1.3 Contador Asíncrono BCD 127
6.1.4 Contador asíncrono binario de 4 bits descendente 129
6.1.5 Contador Asíncrono Binario de 4 bits ascendente / descendente 130
6.2 CONTADOR SíNCRONO 134
6.2.1 Contador Síncrono Binario de 2 bits 134
6.2.2 Contador Síncrono Binario de 4 bits 135
6.3 CONTADOR SíNCRONO ASCENDENTE DESCENDENTE 135
6.4 DISEÑO DE CONTADORES SÍNCRONOS 137
6.5 CONTADORES EN CASCADA 142
Ejercicios Propuestos 145

Capítulo 7. Registros 147
7.1 REGISTROS CON ENTRADA Y SALIDA EN PARALELO 148
7.2 REGISTROS DE DESPLAZAMIENTO CON ENTRADA Y SaLIDA EN SERIE 148
7.3 REGISTROS DE DESPLAZAMIENTO EN CASCADA 150

Ejercicios Propuestos 151

Capítulo 8. Memorias 153
8.1 LECTURA Y ESCRITURA 153
8.1.1 Operación de escritura 154
8.1.2 Operación de lectura 155
8.2 MEMORIAS DE SÓLO LECTURA ROM 155
8.2.1 ROM Básica 156
8.2.2 PROM 156
8.2.3 EPROM 157
8.2.4 UVPROM 157
8.2.5 EEPROM 157
8.3 MEMORIAS DE ACCESO ALEATORIO RAM 158
8.3.1 Arquitectura de RAM Estática (SRAM) 158
8.4 EXPANSIÓN DE MEMORIAS 159
8.4.1 Expansión de longitud de palabra 159
8.4.2 Expansión de capacidad de almacenamiento o tamaño 162
Ejercicios Propuestos 164

BIBLIOGRAFÍA 165
INFOGRAFÍA 165

Índice de figuras

Figura 2.1 Símbolo de la compuerta NOT 46
Figura 2.2 Funcionamiento de la compuerta NOT 46
Figura 2.3 74LS04 Compuerta NOT 46
Figura 2.4 Símbolo de la compuerta AND 48
Figura 2.5 Funcionamiento de la compuerta AND 48
Figura 2.6 74LS08 Compuerta AND de dos entradas 48
Figura 2.7 74LS11 Compuerta AND de tres entradas 49
Figura 2.8 Símbolo de la compuerta OR 50
Figura 2.9 Funcionamiento de la compuerta OR 50
Figura 2.10 74LS32 Compuerta OR de dos entradas 51
Figura 2.11 Símbolo de la compuerta NAND 52
Figura 2.12 Funcionamiento de la compuerta NAND 52
Figura 2.13 74LS00 Compuerta NAND de dos entradas 53
Figura 2.14 74LS10 Compuerta NAND de tres entradas 53
Figura 2.15 Símbolo de la compuerta NOR 54
Figura 2.16 Funcionamiento de la compuerta NOR 55
Figura 2.17 74LS02 Compuerta OR de dos entradas 55
Figura 2.18 74LS27 Compuerta NOR de tres entradas 56
Figura 2.19 Símbolo de la compuerta XOR 57
Figura 2.20 Funcionamiento de la compuerta XOR 57

Figura 2.21 74LS86 Compuerta XOR de dos entradas 58
Figura 2.22 Símbolo de la compuerta XNOR 59
Figura 2.23 Implementación de la compuerta XNOR 59
Figura 2.24 Funcionamiento de la compuerta XNOR 60
Figura 2.25 74LS266 Compuerta XNOR de dos entradas 60
Figura 2.26 Implementación mediante compuertas lógicas 63
Figura 3.1 Ley conmutativa de la adición Booleana 66
Figura 3.2 Ley conmutativa de la multiplicación Booleana 67
Figura 3.3 Ley asociativa de la adición Booleana 67
Figura 3.4 Ley asociativa de la multiplicación Booleana 67
Figura 3.5 Ley distributiva Booleana 68
Figura 3.6 Equivalencias del teorema de DeMorgan 71
Figura 3.7 Implementación de simplificación mediante álgebra de Boole 72
Figura 4.1 Sumador básico 84
Figura 4.2 Sumador completo 85
Figura 4.3 Diagrama en bloques de sumador en paralelo de 4 bits 85
Figura 4.4 74LS283 Sumador en paralelo de 4 bits 86
Figura 4.5 Complemento a 1 y suma para la resta binaria 87
Figura 4.6 Restador de 4 bits 89
Figura 4.7 Comparador de dos cantidades de dos bits 90
Figura 4.8 74LS85 Comparador de dos cantidades de cuatro bits 90
Figura 4.9 Conversor Binario Gray 92
Figura 4.10 Conversor Binario BCD en discreto 94
Figura 4.11 Conversor Binario BCD con sumador 94
Figura 4.12 Decodificador 2-4 95
Figura 4.13 74LS139 Decodificador 2-4 96
Figura 4.14 Decodificador 3-8 97
Figura 4.15 74LS138 Decodificador 3-8 98
Figura 4.16 Display 7 segmentos 99
Figura 4.17 74LS47 Decodificador manejador ánodo común 100
Figura 4.18 Implementación Decodificador Manejador ánodo común 100
Figura 4.19 74LS147 Codificador Decimal BCD de prioridad. 102
Figura 4.20 Multiplexor 4-1 en discreto 104
Figura 4.21 74LS153. Multiplexor 4-1 104
Figura 4.22 Multiplexor 2-1 de 4 bits en discreto 106
Figura 4.23 74LS157. Multiplexor 2-1 de 4 bits 106
Figura 5.1 Latch S-R 109
Figura 5.2 Funcionamiento de Latch S-R 111
Figura 5.3 Latch $\overline{S}-\overline{R}$ 112
Figura 5.4 Funcionamiento de Latch $\overline{S}-\overline{R}$ 113
Figura 5.5 Circuito Antirrebote 114
Figura 5.6 Latch S-R con Habilitación 114
Figura 5.7 Latch D 115
Figura 5.8 Detector de Flancos 116
Figura 5.9 Simbología de circuito con entrada de reloj 117
Figura 5.10 Flip-Flop D 117

Figura 5.11 Flip-Flop J-K 118
Figura 5.12 Símbolo del Flip-Flop J-K 118
Figura 5.13 Flip-Flop J-K con entradas asíncronas 119
Figura 5.14 Símbolo del Flip-Flop J-K con entradas asíncronas 119
Figura 5.15 Diagrama de Temporizador 555 120
Figura 5.16 Diagrama de Temporizador 555 configurado como aestable . 121
Figura 5.17 Divisor de frecuencia 122
Figura 6.1 Contador asíncrono de 2 bits 126
Figura 6.2 Contador asíncrono de 4 bits 126
Figura 6.3 Contador asíncrono BCD 128
Figura 6.4 Contador asíncrono de 4 bits descendente 129
Figura 6.5 Contador asíncrono de 4 bits ascendente descendente 132
Figura 6.6 Contador síncrono de 2 bits 134
Figura 6.7 Contador síncrono de 4 bits 135
Figura 6.8 Contador Ascendente Descendente de 3 bits 137
Figura 6.9 Contador síncrono código Gray de 4 bits 141
Figura 6.10 Contador síncrono código Gray de 4 bits con visualización 142
Figura 6.11 74LS190. Contador síncrono decimal asc/des con carga en paralelo 143
Figura 6.12 74LS190. Contador en cascada módulo 100 144
Figura 7.1 Movimientos de datos en un registro 147
Figura 7.2 Registro con entrada y salida en paralelo 148
Figura 7.3 Registro de desplazamiento hacia la izquierda 149
Figura 7.4 Registro de desplazamiento hacia la derecha 150
Figura 7.5 Registro de desplazamiento en cascada 150
Figura 8.1 Diagrama en bloques de una memoria 154
Figura 8.2 Ilustración de operación de escritura 154
Figura 8.3 Ilustración de operación de lectura 155
Figura 8.4 Tipos de memorias ROM 156
Figura 8.5 Tipos de memorias RAM 158
Figura 8.6 Memoria PROM 32 X 8 160
Figura 8.7 Expansión de longitud de palabra en memoria PROM a 32 X 16160
Figura 8.8 Memoria RAM 1Kb X 8 161
Figura 8.9 Expansión de longitud de palabra en memoria RAM a 1Kb X 16161
Figura 8.10 Expansión de tamaño en memoria PROM a 64 X 8 162
Figura 8.11 Expansión de tamaño en memoria RAM a 2Kb X 8 163

Índice de tablas

Tabla 1.1 Código binario de 2 bits 20
Tabla 1.2 Código binario de 3 bits 21
Tabla 1.3 Equivalencias entre sistemas 22
Tabla 1.4 Código BCD 23
Tabla 1.5 Código Gray 24
Tabla 1.7 Equivalencias binario - hexadecimal 31
Tabla 2.1 Tabla de verdad de la compuerta NOT 45
Tabla 2.2 Tabla de verdad de la compuerta AND de dos entradas 47
Tabla 2.3 Tabla de verdad de la compuerta AND de tres entradas 47
Tabla 2.4 Tabla de verdad de la compuerta OR de dos entradas 49
Tabla 2.5 Tabla de verdad de la compuerta OR de tres entradas 50
Tabla 2.6 Tabla de verdad de la compuerta NAND de dos entradas 51
Tabla 2.7 Tabla de verdad de la compuerta NAND de tres entradas 52
Tabla 2.8 Tabla de verdad de la compuerta NOR de dos entradas 54
Tabla 2.9 Tabla de verdad de la compuerta NOR de tres entradas 54
Tabla 2.10 Tabla de verdad de la compuerta XOR de dos entradas 56
Tabla 2.11 Tabla de verdad de la compuerta XOR de tres entradas 57
Tabla 2.12 Tabla de verdad de la compuerta XNOR de dos entradas 58
Tabla 2.13 Tabla de verdad de la compuerta XNOR de tres entradas 59
Tabla 3.1 Reglas de la suma Booleana 65
Tabla 3.2 Reglas de la multiplicación Booleana 66
Tabla 3.3 Reglas del álgebra de Boole 68
Tabla 3.4 Tabla de verdad del primer teorema de DeMorgan 70
Tabla 3.5 Tabla de verdad del segundo teorema de DeMorgan 70
Tabla 3.6 Simplificación mediante mapa de Karnaugh y tabla de verdad 77
Tabla 3.7 Simplificación de mapa de Karnaugh con valores No importa 78
Tabla 4.1 Tabla de verdad de sumador básico 83
Tabla 4.2 Tabla de verdad de sumador completo 84
Tabla 4.3 Tabla de verdad para el complemento de la resta binaria 88
Tabla 4.4 Tabla de verdad de conversor Binario-Gray 91
Tabla 4.5 Tabla de verdad de conversor Binario-BCD 92
Tabla 4.6 Tabla de verdad del decodificador 2-4 95
Tabla 4.7 Tabla de verdad del decodificador 3-8 96
Tabla 4.8 Tabla de verdad de un decodificador manejador ánodo común 99
Tabla 4.9 Tabla de verdad de un codificador Decimal BCD 101
Tabla 4.10 Tabla de verdad de un codificador Decimal BCD con prioridad 102
Tabla 4.11 Tabla de verdad de un multiplexor 4-1 103
Tabla 4.12 Tabla de verdad de un multiplexor 2-4 de 4 bits 105
Tabla 5.1 Tabla de verdad de Latch S-R 111
Tabla 5.2 Tabla de verdad de Latch $\overline{S}-\overline{R}$ 113

Tabla 5.3 Tabla de verdad de Latch S-R con Habilitación 115
Tabla 5.4 Tabla de verdad de Latch D 115
Tabla 5.5 Tabla de verdad de Flip-Flop D 117
Tabla 5.6 Tabla de verdad de Flip-Flop J-K 118
Tabla 6.1 Tabla de secuencia de contador asíncrono de 2 bits 126
Tabla 6.2 Tabla de secuencia de contador asíncrono de 4 bits 127
Tabla 6.3 Tabla de secuencia de contador asíncrono BCD de 4 bits 128
Tabla 6.4 Tabla de secuencia de contador asíncrono de 4 bits descendente 130
Tabla 6.5 Tabla de control para contador asíncrono ascendente-descendente 130
Tabla 6.6 Tabla de secuencia de contador asíncrono de 4 bits ascendente-descendente 133
Tabla 6.7 Tabla de secuencia de contador asíncrono de 4 bits ascendente-descendente 136
Tabla 6.8 Tabla de secuencia de contador código Gray 138
Tabla 6.9 Tabla de transición de estados del Flip-Flop J-K 138

INTRODUCCIÓN

Diseño Lógico ofrece al lector una exposición clara y suficiente de los conceptos básicos de los sistemas digitales combinacionales y secuenciales.

En esta obra se puede obtener el conocimiento y habilidad necesaria para resolver diseños de electrónica digital con base en los fundamentos del mismo.

El documento evidencia una exposición de los conceptos de la misma forma en que éstos han venido evolucionando. Con base en ello, es importante tener en cuenta que cada uno de los conceptos presentados depende ampliamente de los conceptos anteriores. De esta forma se llega a la comprensión total de cada uno de los temas.

La lógica combinacional trata dispositivos con una característica fundamental que consiste en que cada salida de un circuito lógico depende totalmente de la combinación lógica de entrada que se le aplique. Dentro de estos dispositivos están los sumadores, codificadores, multiplexores, entre otros. La lógica secuencial trata dispositivos en donde su salida depende de una señal digital temporizada que se obtiene a través de un oscilador digital que actúa a una frecuencia deseada. Esta característica permite que los dispositivos lógicos secuenciales (los contadores, registros, memorias, etc.) adquieran la capacidad de almacenamiento de información.

GLOSARIO

Acarreo. Bit generado cuando la suma de dos números binarios excede la base en 1.

Aestable. Refiere a un comportamiento de una señal sin un estado estable. El multivibrador aestable oscila a una frecuencia dada entre dos estados.

AND. Operación lógica que coloca en la salida el valor verdadero o uno lógico si todas sus entradas tienen valor verdadero.

Asíncrono. Circuito que no posee relación temporal fija. En un contador, refiere a la presencia de relojes independientes entre Flip-Flops.

Basculación. Acción de un Flip-Flop de cambio de estado con cada flanco de reloj.

BCD Binary Coded Decimal. Código binario que representa únicamente los dígitos decimales.

Binario. Sistema numérico con valores 0 o 1.

Bit. Símbolo binario.

Byte. Conjunto de 8 bits.

Cascada. Conexión de dispositivos uno tras otro.

Circuito Combinacional. Circuito digital donde los estados lógicos de las salidas dependen estrictamente de los estados lógicos de las entradas.

Circuito Integrado. Circuito en donde los componentes se encuentran integrados en un chip semiconductor.

Circuito Secuencial. Circuito digital donde los estados lógicos de las salidas dependen de una señal de reloj.

Codificador de prioridad. Codificador que codifica el digito de entrada de valor más alto.

Código Gray. Código binario caracterizado por el cambio de un único bit entre valores codificados adyacentes.

Contador. Circuito secuencial, capaz de contar estados de acuerdo con una secuencia previamente establecida.

Contador Asíncrono. Contador en donde los relojes de los Flip-Flops son diferentes.

Contador Síncrono. Contador en donde los relojes de los Flip-Flops son el mismo.

Conversor de Código. Circuito combinacional que permite convertir información codificada de una manera a otra.

Demultiplexor. Circuito combinacional que conmuta datos digitales desde una línea de entrada a varias líneas de salida según una secuencia definida.

Digito. Símbolo del sistema decimal.

Diodo. Dispositivo semiconductor que permite el paso de corriente en una sola dirección denominada polarización directa.

DRAM. Dynamic Random Access Memory. Memoria de acceso aleatorio dinámica. Memorial de lectura y escritura volátil.

EPROM. Eraseble Programmable Read Only. Memoria de solo lectura programable borrable.

EEPROM. Electrically Eraseble Programmable Read Only. Memoria de solo lectura programable borrable eléctricamente.

Flip-Flop. Circuito secuencial con capacidad de almacenamiento de información de un solo bit.

Flip-Flop D. Tipo de Flip-Flop en donde su salida Q corresponde a su entrada D.

Flip-Flop J-K. Tipo de Flip-Flop que contiene los estados de No Cambio, Set, Reset y Basculación.

Frecuencia. Numero de ciclos de una señal en un segundo

Glitch. Flanco no deseado producido de forma no intencionada.

Hexadecimal. Sistema numérico de base 16.

Latch. Dispositivo biestable capaz de almacenar un bit.

LED. Light Emisor Diode. Diodo emisor de luz. Diodo que emite fotones en configuración de polarización directa.

Ley Asociativa. En la suma o multiplicación de tres o más variables, el orden en que se agrupan no altera el resultado

Ley Conmutativa. En la suma o multiplicación de dos variables, el orden de los valores no altera el resultado

Ley Distributiva. Al sumar varias variables y luego multiplicar el resultado por una sola variable, es equivalente a multiplicar la variable aislada con cada una de la variables operadas mediante la suma.

Lógica combinacional. Combinación de compuertas lógicas para implementar una expresión booleana que no requiera almacenamiento de información.

Mapa de Karnaugh. Matriz que permite simplificación de expresión booleana mediante la representación de valores binarios por cada celda de la matriz.

Multiplexor. Circuito combinacional que conmuta datos digitales de distintas líneas de entrada a una única línea de salida según una secuencia definida.

Negativa-AND. Operación equivalente a la operación lógica NOR.

Negativa-OR. Operación equivalente a la operación lógica NAND.

Nibble. Conjunto de 4 bits.

NOT. Operación lógica que realiza la inversión de la entrada binaria.

Octal. Sistema numérico de base 8.

OR. Operación lógica que coloca en la salida el valor false o cero lógico si todas sus entradas tienen valor false.

OR Exclusiva XOR. Operación lógica presenta a la salida valor verdadero si el número de entradas verdaderas es impar.

RAM. Random Access Memory. Memoria de acceso aleatorio.

Registro. Circuito secuencial capaz de almacenar y desplazar información binaria.

Reloj. Señal de temporización

RESET. Estado de un Flip-Flop que coloca la salida Q en cero.

Restador. Circuito combinacional que realiza la resta entre dos cantidades binarias.

ROM. Read Only Memory. Memoria de solo lectura.

SET. Estado de un Flip-Flop que coloca la salida Q en uno.

Restador. Circuito combinacional que realiza la suma entre dos cantidades binarias.

Tabla de Verdad. Tabla que muestra las entradas y salidas con sus correspondientes valores de una expresión booleana.

Volátil. Término con que se describe la característica de una memoria de pérdida de información en ausencia de energía.

Capítulo 1
Sistemas Numéricos

Los sistemas numéricos son un concepto fundamental para el estudio de la electrónica digital. Dentro de los sistemas numéricos que se pretenden utilizar se encuentran el sistema decimal, el sistema octal, el sistema binario y el sistema hexadecimal. Cada uno de estos sistemas tiene una base, la cual indica la cantidad de símbolos del sistema.

1.1 SISTEMA DECIMAL

El sistema numérico decimal, es un sistema base 10 debido a que tiene 10 símbolos los cuales son los números del 0 al 9. Para este sistema a cada símbolo se le denomina ***dígito*** y es el más comúnmente usado en la vida cotidiana, ya que por medio de éste podemos representar cualquier cantidad numérica estándar. Cuando se trabaja números que se encuentran en diferentes sistemas, es aconsejable identificar la base del número colocándola como subíndice. Por ejemplo, si se desea representar el número 25 en decimal se coloca: 25_{10}

Al tener 10 dígitos, se pueden formar números de diferentes cantidades. Un número con una sola cifra, tiene 10 posibles cantidades que van de 0 a 9. Si un número tiene dos cifras, entonces se pueden tener hasta 100 posibles cantidades.

Cada cifra dentro de un número tiene un valor que es comúnmente llamado como ***peso.*** La cifra de menor peso, siempre es la cifra ubicada a la derecha del número. Los pesos de cada cifra incrementan de forma exponencial en donde la base es la cantidad de símbolos del sistema y el exponente es la posición de cada símbolo.

Entonces la cantidad de un número se obtiene de la siguiente forma:

$$348 = 8 * 10^0 + 4 * 10^1 + 3 * 10^2$$
$$348 = 8 * 1 + 4 * 10 + 3 * 100$$
$$348 = 8 + 40 + 300$$

La posición de cada dígito en un número decimal indica la magnitud de la cantidad representada. Los pesos para los números enteros con potencias positivas de diez que aumentan de derecha a izquierda comenzando por 10^0. Para los números fraccionarios, los pesos son potencias negativas de diez que aumentan de izquierda a derecha comenzando por 10^{-1}.

... 10^2 10^1 10^0 . 10^{-1} 10^{-2} ...

Por ejemplo, la representación del número decimal 568,25 como suma de valores de cada dígito es:

$$568{,}25 = 5 * 10^2 + 6 * 10^1 + 8 * 10^0 + 2 * 10^{-1} + 5 * 10^{-2}$$
$$568{,}25 = 5 * 100 + 6 * 100 + 8 * 100 + 2 * 0{,}1 + 5 * 0{,}01$$
$$568{,}25 = 500 + 60 + 8 + 0{,}2 + 0{,}05$$

1.2 SISTEMA BINARIO

El sistema numérico binario es un sistema base 2 debido a que tiene 2 símbolos los cuales son los números 0 y 1. En este sistema a un símbolo se le denomina ***bit***. Es aconsejable identificar la base del número colocándola como subíndice. Por ejemplo, si se desea representar el número 100110 en binario se coloca: 100110_2

Para formar cantidades binarias es necesario tener en cuenta que sólo hay 2 símbolos en el sistema. Con base en esto, un número con una sola cifra solo puede tener 2 posibles combinaciones. Un número con 2 cifras puede tener 4 posibles combinaciones. Esto es debido a que los pesos para los números binarios se calculan con potencias positivas de base 2 que aumentan de derecha a izquierda comenzando por 2^0.

Cada número binario tiene su equivalente decimal. Por ejemplo si se tiene números binarios de 2 bits se pueden lograr las siguientes combinaciones:

Tabla 1.1 Código binario de 2 bits

Decimal	Binario
0	00
1	01
2	10
3	11

Con 3 bits se puede lograr las siguientes combinaciones:

Tabla 1.2 Código binario de 3 bits

Decimal	Binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Entonces se puede observar que la cantidad de combinaciones que se pueden obtener en binario corresponde a 2^n donde n es el número de bits. Con base en lo anterior, con 4 bits se pueden obtener 16 combinaciones, con 5 bits 32 combinaciones y así sucesivamente.

Igual que el sistema decimal, cada bit tiene un peso. Los pesos para los números binarios son potencias positivas base dos que aumentan de derecha a izquierda comenzando por 2^0.

... 2^2 2^1 2^0

1.3 SISTEMA OCTAL

El sistema numérico octal es un sistema base 8 debido a que tiene 8 símbolos los cuales son los números del 0 al 7. Es necesario identificar la base del número colocándola como subíndice. Por ejemplo, si se desea representar el número 256 en octal se coloca: 256_8. La cantidad de combinaciones que se pueden obtener en octal corresponde a 8^n donde n es el número de símbolos.

Igual que el sistema decimal, cada símbolo tiene un peso. Los pesos para los números octales son potencias positivas base ocho que aumentan de derecha a izquierda comenzando por 8^0.

... 8^2 8^1 8^0

1.4 SISTEMA HEXADECIMAL

El sistema numérico hexadecimal es un sistema base 16 debido a que tiene 16 símbolos los cuales son los números del 0 al 9 y de la letra A a la F. Es nece-

sario identificar la base del número colocándola como subíndice. Por ejemplo, si se desea representar el número 95B en binario se coloca: $95B_{16}$. La cantidad de combinaciones que se pueden obtener en hexadecimal corresponde a 16^n donde n es el número de símbolos.

Igual que el sistema decimal, cada símbolo tiene un peso. Los pesos para los números hexadecimales son potencias positivas base dieciséis que aumentan de derecha a izquierda comenzando por 16^0.

... 16^2 16^1 16^0

Cada símbolo en hexadecimal tiene una equivalencia en los demás sistemas que se presentan en la siguiente tabla:

Tabla 1.3 Equivalencias entre sistemas

Decimal	Binario	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.5 CÓDIGO BCD

Vale la pena mencionar el concepto de código BCD en este capítulo, debido a que este código tiene fuerte relación con el sistema binario y el sistema decimal.

BCD es la sigla de *Binary Coded Decimal.* Significa Decimal codificado binario. Este código tiene una única función que es representar únicamente los símbolos decimales en binario. Con base en lo anterior, el código BCD no es equivalente a código base 2. El código BCD es el de la siguiente tabla:

Tabla 1.4 Código BCD

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Entonces si se desea representar un valor decimal de más de un dígito en BCD, se toma el código BCD de cada dígito y se colocan en el peso correspondiente.

Por ejemplo, si se desea representar el número 45_{10} en BCD quedaría 0100 0101. El 0100_2 equivale a 4_{10} y el 0101_2 equivale a 5_{10}.

1.6 CÓDIGO GRAY

El código Gray es un código con una característica fundamental. La variación entre una combinación y otra es de solo un bit. Esta característica es bastante importante en muchas aplicaciones como detección y corrección de errores en sistemas de comunicaciones digitales, de allí la importancia del mismo.

Para construir el código se debe iniciar con los símbolos del código binario en el bit de menor peso, es decir con el 0 y 1. Después se refleja el código resultante, es decir, que se continúa el código colocando el 1 y 0, luego se coloca el bit de siguiente peso con la mitad de posiciones en 0 y la otra mitad en 1.

00
01
11
10

Después de esto se vuelve a reflejar el código obteniendo:

000
001
011
010
110
111
101
100

Finalmente se obtiene el código Gray (tabla 1.5) de 4 bits.

Tabla 1.5 Código Gray

Binario	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

1.7 CONVERSIÓN ENTRE SISTEMAS

Es posible convertir un número que se encuentra en un sistema a cualquier otro sistema. Es decir, si tenemos un número decimal, podemos representarlo en binario, octal y hexadecimal.

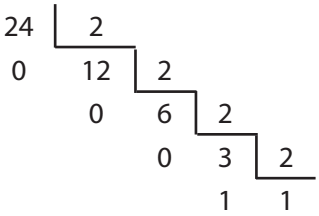
1.7.1 Conversión Decimal-Binario

Esta conversión consiste en obtener un número binario equivalente de un número decimal. Hay dos métodos comúnmente conocidos.

Método de divisiones sucesivas

Este método consiste en tomar el número decimal y dividirlo en 2 (debido a que la base del sistema binario es 2). La división se repite hasta que el cociente sea menor que el divisor. Para este caso en particular, debido a que el divisor es 2 entonces se hace divisiones hasta que el cociente sea 1. Al finalizar el número binario se construye tomando el último cociente y se toma los residuos de las divisiones desde el último hasta el primer residuo.

Por ejemplo si tomamos el número decimal 24_{10} le aplicamos el método quedaría de la siguiente forma:



Finalmente el valor binario sería 11000_2.

Método de suma de pesos

Este método consiste en calcular el número binario equivalente al número decimal dado mediante la suma de los pesos binarios que dan como resultado el número decimal. Es necesario tener en cuenta que los pesos binarios van de 2^0 hasta 2^n donde n es el número de bits. Esto es equivalente a decir que los pesos son: 1, 2, 4, 8, 16, 32, etc.

Entonces, una forma práctica de aplicar el algoritmo es encontrar el peso mayor en binario para el número dado.

Por ejemplo, si tenemos el número 25_{10} entonces debemos encontrar un peso que no exceda esa cantidad. Si se escoge el peso 5, $2^5 = 32$ y excede el 25. Entonces el peso adecuado a escoger es el 4.

$$2^4 = 16$$
$$25 - 16 = 9$$

Aplicamos el mismo algoritmo para el valor resultante que es el 8.

$$2^3 = 8$$
$$9 - 8 = 1$$

Aplicamos el mismo algoritmo para el valor resultante que es el 1. Nos damos cuenta que $2^2 = 4$ excede el 1, $2^1 = 2$ excede el 1, entonces:

$$2^0 = 1$$
$$1 - 1 = 0$$

Finalmente nos damos cuenta que debe intervenir el peso 4, 3 y 0. Eso significa que para estos pesos el valor es 1. Para los pesos 2 y 1 el valor debe ser 0.

$$\begin{array}{ccccc} 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 1 & 1 & 1 & 1 \end{array}$$
$$16 + 8 + 0 + 0 + 1 = 25$$

Entonces el número binario equivalente al número 25 decimal es 11001_2.

1.7.2 Conversión Binario-Decimal

Esta conversión consiste en obtener un número decimal equivalente de un número binario.

Método de suma de pesos

Este método consiste en calcular el valor de cada peso del número binario y hacer la sumatoria de sus resultados. El resultado de la sumatoria equivaldrá al número decimal.

Por ejemplo si tenemos el número 11011010_2 hacemos las siguientes operaciones:

1	1	0	1	1	0	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
$128 * 1 = 128$	$64 * 1 = 64$	$32 * 0 = 0$	$16 * 1 = 16$	$8 * 1 = 8$	$4 * 0 = 0$	$2 * 1 = 2$	$1 * 0 = 0$

$$128 + 64 + 16 + 8 + 2 = 218$$

1.7.3 Conversión Decimal-Octal

Esta conversión consiste en obtener un número octal equivalente de un número decimal. Hay dos métodos comúnmente conocidos.

Método de divisiones sucesivas

Este método consiste en tomar el número decimal y dividirlo en 8 (debido a que la base del sistema octal es 8). La división se repite hasta que el cociente sea menor que el divisor. Al finalizar el número octal se construye tomando el último cociente y se toma los residuos de las divisiones desde el último hasta el primer residuo.

Por ejemplo si tomamos el número decimal 95_{10} le aplicamos en método quedaría de la siguiente forma:

95	8	
7	11	8
	3	1

Finalmente el valor octal sería 137_8.

Método de suma de pesos

Este método consiste en calcular el número octal equivalente al número decimal dado mediante la suma de los pesos octales que dan como resultado el número decimal. Es necesario tener en cuenta que los pesos octales van de 8^0 hasta 8^n donde n es el número de cifras del número octal. Esto es equivalente a decir que los pesos son: 1,8,64,512, etc. El peso de cada símbolo se debe multiplicar por el símbolo correspondiente. Ese resultado es el que se sumará para obtener el número octal.

Entonces, una forma práctica de aplicar el algoritmo es encontrar el peso mayor en octal para el número dado.

Por ejemplo si tenemos el número 95_{10} entonces debemos encontrar un peso que multiplicado por un símbolo octal, no exceda esa cantidad. Si se escoge el peso 3, 8^3=512 y excede el 95, entonces el peso adecuado a escoger es el 2.

$$8^2 = 64$$
$$64 * 1 = 64$$
$$95 - 64 = 9 = 3$$

Aplicamos el mismo algoritmo para el valor resultante que es el 31.

$$8^1 = 8$$
$$8 * 3 = 24$$
$$31 - 24 = 7$$

Aplicamos el mismo algoritmo para el valor resultante que es el 7.

$8^0 = 1$
$1 * 7 = 7$
$7 - 7 = 0$

Entonces:

$8^2 \quad 8^1 \quad 8^0$
$64 \quad 8 \quad 1$
$64 * 1 + 8 * 3 + 1 * 7 = 95$

Entonces el número octal equivalente al número 95 decimal es 137_8.

1.7.4 Conversión Octal-Decimal

Esta conversión consiste en obtener un número decimal equivalente de un número octal.

Método de suma de pesos

Este método consiste en calcular el valor de cada peso del número octal, multiplicarlo por la cifra octal correspondiente y hacer la sumatoria de sus resultados. El resultado de la sumatoria equivaldrá al número decimal.

Por ejemplo si tenemos el número 7315_8 hacemos las siguientes operaciones:

7	3	1	5
8^3	8^2	8^1	8^0
$512 * 7 = 3584$	$64 * 3 = 192$	$8 * 1 = 8$	$1 * 5 = 5$

$3584 + 192 + 8 + 5 = 3789$

1.7.5 Conversión Decimal-Hexadecimal

Esta conversión consiste en obtener un número hexadecimal equivalente de un número decimal.

Método de divisiones sucesivas

Este método consiste en tomar el número decimal y dividirlo en 16 (debido a que la base del sistema hexadecimal es 16). La división se repite hasta que el cociente sea menor que el divisor. Al finalizar el número hexadecimal se construye tomando el último cociente y se toma los residuos de las divisiones desde el último hasta el primer residuo. Se debe tener en cuenta que el último cociente o residuos pueden ser números decimales de 10 a 15. Para estos casos, se hace la equivalencia mostrada en la tabla 1.3.

Por ejemplo si tomamos el número decimal 200_{10} le aplicamos el método quedaría de la siguiente forma:

200	16
8	12

Siguiendo el algoritmo, se toma el último cociente y se toma los residuos desde el último hasta el primero. Entonces el valor hexadecimal sería $C8_{16}$.

Método de suma de pesos

Este método consiste en calcular el número hexadecimal equivalente al número decimal dado mediante la suma de los pesos hexadecimales que dan como resultado el número decimal. Es necesario tener en cuenta que los pesos hexadecimales van de 16^0 hasta 16^n donde n es el número de cifras del número. Esto es equivalente a decir que los pesos son: 1,16,256, etc. El peso de cada símbolo se debe multiplicar por el símbolo del mismo peso. Ese resultado es el que se sumará para obtener el número hexadecimal.

Entonces, una forma práctica de aplicar el algoritmo es encontrar el peso mayor en hexadecimal para el número dado.

Por ejemplo si tenemos el número 200_{10} entonces debemos encontrar un peso que multiplicado por un símbolo octal, no exceda esa cantidad. Si se escoge el peso 2, 16^2=256 y excede el 200. Entonces el peso adecuado a escoger es el 1.

$$16^1 = 16$$
$$16 * 12 = 192$$
$$200 - 192 = 8$$

Aplicamos el mismo algoritmo para el valor resultante que es el 8.

$$16^0 = 1$$
$$1 * 8 = 8$$
$$8 - 8 = 0$$

Entonces:

16^1	16^0
12	8

$$16 * 12 + 1 * 8 = 200$$

Teniendo en cuenta que los valores resultantes se encuentran entre 10 y 15 se realiza la equivalencia, entonces el resultado es $C8_{16}$.

1.7.6 Conversión Hexadecimal-Decimal

Esta conversión consiste en obtener un número decimal equivalente de un número hexadecimal.

Método de suma de pesos

Este método consiste en calcular el valor de cada peso del número hexadecimal multiplicarlo por la cifra octal correspondiente y hacer la sumatoria de sus resultados. El resultado de la sumatoria equivaldrá al número decimal.

Por ejemplo, si tenemos el número $45B6_{16}$ hacemos las siguientes operaciones

$$\begin{array}{cccc} 4 & 5 & B & 6 \\ 16^3 & 16^2 & 16^1 & 16^0 \\ 4096 * 4 = 16384 & 256 * 5 = 1280 & 16 * 11 = 176 & 1 * 6 = 6 \end{array}$$

$$16384 + 1280 + 176 + 6 = 17846$$

1.7.7 Conversión Binario-Octal

Esta conversión consiste en obtener un número octal equivalente de un número binario.

El método indica que se debe hacer grupos de tres bits en el número binario, de derecha a izquierda. Cada grupo de tres bits representa un símbolo octal, debido a que el sistema octal tiene 8 símbolos y con 3 bits se pueden tener 8 combinaciones. Si el último grupo de la izquierda no tiene 3 bits, se rellena con ceros a la izquierda. Se puede entonces utilizar la siguiente tabla de equivalencia.

Tabla 1.6. Equivalencias binario - octal

Binario	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Con base en lo anterior, si se tiene el número binario 1001010111_2 el equivalente octal sería:

001	001	010	111
1	1	2	7

Entonces el número octal es 1127_8

1.7.8 Conversión Octal-Binario

Esta conversión consiste en obtener un número binario equivalente de un número octal.

Cada símbolo octal equivale a un valor binario de 3 bits, debido a que el sistema octal tiene 8 símbolos y con 3 bits se pueden tener 8 combinaciones. Se puede utilizar la tabla 1.6 como equivalencias.

Con base en lo anterior, si se tiene el número octal 7516_8 el equivalente binario sería:

7	5	1	6
111	101	001	110

Entonces el número binario es 111101001110_2

1.7.9 Conversión Binario-Hexadecimal

Esta conversión consiste en obtener un número hexadecimal equivalente de un número binario.

El método indica que se debe hacer grupos de cuatro bits en el número binario, de derecha a izquierda. Cada grupo de cuatro bits representa un símbolo hexadecimal, debido a que el sistema hexadecimal tiene 16 símbolos y con 4 bits se pueden tener 16 combinaciones. Si el último grupo de la izquierda no tiene 4 bits, se rellena con ceros a la izquierda. Se puede entonces utilizar la siguiente tabla de equivalencia.

Tabla 1.7 Equivalencias binario - hexadecimal

Binario	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4

0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Con base en lo anterior, si se tiene el número binario 1101110010111_2 el equivalente hexadecimal sería:

0001	1011	1001	0111
1	*B*	9	7

Entonces el número hexadecimal es $1B97_{16}$

1.7.10 Conversión Hexadecimal-Binario

Esta conversión consiste en obtener un número binario equivalente de un número hexadecimal.

Cada símbolo hexadecimal equivale a un valor binario de 4 bits, debido a que el sistema hexadecimal tiene 16 símbolos y con 4 bits se pueden tener 16 combinaciones. Se puede utilizar la tabla 1.7 como equivalencias

Con base en lo anterior, si se tiene el número octal $B45A_{16}$ el equivalente binario sería:

B	4	5	*A*
1011	0100	0101	1010

Entonces el número binario es 1011010001011010_2

1.8 COMPLEMENTO A 1 Y COMPLEMENTO A 2

El complemento a 1 y complemento a 2 de números representados en sistemas diferentes al decimal, es supremamente importante, porque gracias a este concepto se puede representar números negativos. Estos conceptos son usados en computación para hacer operaciones aritméticas con números negativos.

1.8.1 Complemento a 1

El complemento a 1, consiste en obtener el valor que le hace falta a un símbolo para llegar al símbolo máximo de la base. Para el caso particular del sistema binario, el complemento a la base consiste en cambiar ceros por unos y unos por ceros, debido a que solo hay estos dos símbolos en el sistema.

El complemento a 1 del número binario 101101010_2 sería:

101101010
010010101

El complemento a 1 del número octal 27610_8 sería:

27610
50167

El complemento a 1 del número hexadecimal $A4B2_{16}$ sería:

A4B2
5B4D

El complemento a 1 del número decimal 846_{10} sería:

846
153

1.8.2. Complemento a 2

El complemento a 2, consiste en obtener el complemento a 1 y sumarle 1.

El complemento a 1 del número binario 101101010_2 sería:

101101010
010010101 + 1 = 010010110

El complemento a 1 del número octal 27610_8 sería:

27610
50167 + 1 = 50170

El complemento a 1 del número hexadecimal $A4B2_{16}$ sería:

A4B2
5B4D + 1 = *5B4E*

El complemento a 1 del número decimal 846_{10} sería:

846
153 + 1 = 154

1.9 OPERACIONES ARITMÉTICAS DE DIFERENTES SISTEMAS

Las operaciones aritméticas son fundamentales para cualquier tipo de proceso en las máquinas. Dentro de las operaciones aritméticas más comunes se encuentran la ***suma*** y ***resta***. Otras operaciones aritméticas que se basan además de la suma y resta son la ***multiplicación y división***.

En este capítulo se trabajarán las operaciones aritméticas suma y resta para los sistemas binario, octal y hexadecimal y se trabajarán las operaciones multiplicación y división únicamente para el sistema binario, debido a que para hacer multiplicación y división en otros sistemas es necesario conocer las tablas de multiplicar de dichos sistemas.

1.9.1 Suma en Binario

Existen unas reglas básicas para realizar la suma en binario. Como el sistema binario tiene dos símbolos, entonces sólo existen cuatro posibles sumas con dos cantidades de un bit. Ellas son:

0 + 0 = 0 con acarreo = 0
0 + 1 = 1 con acarreo = 0
1 + 0 = 1 con acarreo = 0
1 + 1 = 1 con acarreo = 1

El acarreo es un valor que debe ser utilizado en el siguiente peso del número binario.

La suma binaria de los sumandos $101101_2 + 1011_2$ es:

$$\begin{array}{cccccc} & 1 & 1 & 1 & 1 & \\ 1 & 0 & 1 & 1 & 0 & 1 \\ & & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

En la suma del bit de menor peso, se tiene 1+1, el resultado es 0 con un acarreo que se coloca en el siguiente peso, dando como resultado total 111000_2

1.9.2 Suma en Octal

Para hacer la suma en octal es importante tener en cuenta que existen solo 8 símbolos que van del 0 al 7. Entonces si una suma, supera el 7, existe entonces un acarreo debido a que se excede la base. Por ejemplo si se tiene 5 + 6 enton-

ces a 5 se le agrega 6, sin embargo si a 5 se le agrega 2, llega al valor máximo de la base, si se le agrega 3, existe un acarreo y el resultado sería entonces 0. Por consiguiente 5 + 6 = 13, donde 1 es el acarreo y 3 el resultado.

Se puede representar también de la siguiente forma:

- Tomamos los símbolos del sistema octal
  0 1 2 3 4 5 6 7
- Nos ubicamos en el primer sumando que es 5
  0 1 2 3 4 $\underline{5}$ 6 7
- Adicionamos el segundo sumando
  3 4 5 6 1 2
  0 1 2 3 4 $\underline{5}$ 6 7

Se puede notar que al sumarle a 5 un 6, se excede la base y el resultado es 3, por consiguiente la suma da como resultado 13_8

La suma octal de los sumandos $3462_8 + 413_8$ es:

$$\begin{array}{cccc} 1 & & & \\ 3 & 4 & 6 & 2 \\ & 4 & 1 & 3 \\ \hline 4 & 0 & 7 & 5 \end{array}$$

El resultado es 4075_8

1.9. 3 Suma en hexadecimal

Para hacer la suma en hexadecimal es importante tener en cuenta que existen 16 símbolos que van del 0 al 9 y de la A a la F. Entonces si una suma, supera la F, existe entonces un acarreo debido a que se excede la base. Por ejemplo si se tiene A + 8 entonces a A se le agrega 8. Por consiguiente A + 8 = 12, donde 1 es el acarreo y 2 el resultado.

Se puede representar también de la siguiente forma:

- Tomamos los símbolos del sistema hexadecimal
  0 1 2 3 4 5 6 7 8 9 *A B C D E F*
- Nos ubicamos en el primer sumando que es A
  0 1 2 3 4 5 6 7 8 9 $\underline{A}$ *B C D E F*
- Adicionamos el segundo sumando
  6 7 8 1 2 3 4 5
  0 1 2 3 4 5 6 7 8 9 $\underline{A}$ *B C D E F*

Se puede notar que al sumarle a A un 8, se excede la base y el resultado es 2, por consiguiente la suma da como resultado 12_{16}

La suma octal de los sumandos $A463_{16}$ + 13E16 es:

$$\begin{array}{cccc} & & 1 & \\ A & 4 & 6 & 3 \\ & 1 & 3 & E \\ \hline A & 5 & A & 1 \end{array}$$

El resultado es $A4A1_{16}$

1.9.4 Resta en Binario

Para realizar la resta en binario se requiere seguir un algoritmo descrito por los siguientes pasos:

- Se iguala el número de cifras en el minuendo y el sustraendo añadiendo ceros a la izquierda del número con menos cifras.
- Se conserva el minuendo.
- Se complementa a 1 el sustraendo.
- Se suma.
- Si existe acarreo, se suma el acarreo. La existencia del acarreo indica que el resultado es positivo.
- Si no existe acarreo, se complementa a 1 el resultado. La no existencia de acarreo indica que el resultado es negativo.

Con base en la anterior descripción, la resta $10110101_2 - 10001010_2$ es:

$$\begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ - \; 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array}$$

$$\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & & 1 & & 1 & & \\ & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ + & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ \hline \underline{1} & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ & & & & & & & + & \underline{1} \\ \hline & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{array}$$

El resultado es 101011_2 positivo

Ahora por ejemplo la resta $10110101_2 - 11001010_2$ es:

$$\begin{array}{r} 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\ -\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0 \\ \hline \end{array}$$

$$\begin{array}{r} 0\ 1\ 1\ \ \ 1\ \ \ 1 \\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\ +\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\ \hline 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0 \\ -\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \end{array}$$

El resultado es 10101_2 negativo

1.9.5 Resta en octal

Para realizar la resta en octal se aplica el mismo algoritmo de resta binaria.

Entonces, la resta $7156_8 - 4326_8$ es:

$$\begin{array}{r} 7\ 1\ 5\ 6 \\ -\ 4\ 3\ 2\ 6 \\ \hline \end{array}$$

$$\begin{array}{r} 1\ \ \ 1 \\ 7\ 1\ 5\ 6 \\ +\ 3\ 4\ 5\ 1 \\ \hline \underline{1}\ 2\ 6\ 2\ 7 \\ \qquad\quad +\ \underline{1} \\ \hline 2\ 6\ 3\ 0 \end{array}$$

El resultado es 2630_8 positivo

Ahora por ejemplo la resta $3120_8 - 4033_8$ es:

$$\begin{array}{r} 3\ 1\ 2\ 0 \\ -\ 4\ 0\ 3\ 3 \\ \hline \end{array}$$

$$\begin{array}{r} 1 \\ 3\ 1\ 2\ 0 \\ +\ 3\ 7\ 4\ 4 \\ \hline 0\ 7\ 0\ 6\ 4 \\ -\ 0\ 7\ 1\ 3 \end{array}$$

El resultado es 713_8 negativo

1.9.6 Resta en hexadecimal

Para realizar la resta en octal se aplica el mismo algoritmo de resta binaria.

Entonces, la resta $A4B6_{16} - 4CD6_{16}$ es:

$$\begin{array}{rcccc} & A & 4 & B & 6 \\ - & 4 & C & D & 6 \\ \hline \end{array}$$

$$\begin{array}{rcccc} 1 & & & & \\ & A & 4 & B & 6 \\ + & B & 3 & 2 & 9 \\ \hline 1 & 5 & 7 & D & F \\ & & & + & 1 \\ \hline & 5 & 7 & E & 0 \end{array}$$

El resultado es $57E0_{16}$ positivo

Ahora por ejemplo la resta $B84_{16} - C1B3_{16}$ es:

$$\begin{array}{rcccc} & B & 8 & 4 & \\ - & C & 1 & B & 3 \\ \hline \end{array}$$

$$\begin{array}{rcccc} & 1 & & 1 & \\ & & B & 8 & 4 \\ + & 3 & E & 4 & C \\ \hline 0 & 4 & 9 & D & 0 \\ - & B & 6 & 2 & F \end{array}$$

El resultado es $B62F_{16}$ negativo

1.9.7 Multiplicación en binario

Existen dos algoritmos para realizar la multiplicación en binario. Un algoritmo se basa en el algoritmo de multiplicación decimal y otro algoritmo se basa en rotaciones. Para cualquiera de los dos algoritmos es necesario conocer las tablas de multiplicar en binario, éstas son:

$$0 * 0 = 0$$
$$0 * 1 = 0$$
$$1 * 0 = 0$$
$$1 * 1 = 1$$

Algoritmo de multiplicación decimal para binario

Dado un multiplicando binario y un multiplicador binario, la multiplicación se realiza multiplicando el bit de menor peso del multiplicador con el multiplicando. Se realiza el algoritmo para todos los bits del multiplicador. Finalmente se realiza la suma de los resultados.

Por ejemplo la multiplicación de 110101_2 * 10011 es:

					1	1	0	1	0	1
				X		1	0	0	1	1
					1	1	0	1	0	1
				1	1	0	1	0	1	
			0	0	0	0	0	0		
		0	0	0	0	0	0			
1	1	0	1	0	1					
1	1	1	1	1	0	1	1	1	1	

El resultado es 1111101111_2

Algoritmo de multiplicación binario por rotaciones

Dado un multiplicando binario y un multiplicador binario, la multiplicación por rotaciones procede con el siguiente algoritmo:

Se inicializa el resultado con 0 binario (La cantidad de bits, depende de la cantidad de bits mayor entre el multiplicando y multiplicador).

Se toma el primer bit del multiplicador de izquierda a derecha.

Si el bit tomado es 0, se debe rotar el resultado a la izquierda (para este caso en particular, rotar a la izquierda es equivalente a agregar un cero a la derecha).

Si el bit tomado es 1, se debe rotar el resultado a la izquierda y sumar el multiplicando.

Se repite el proceso desde el paso 2 hasta el 4 para el siguiente bit del multiplicador hasta el último bit de la derecha del multiplicador.

Por ejemplo la multiplicación de 110101_2 * 10011 es:

Primer paso: se inicializa el resultado con 0. Se toma el primer bit de derecha a izquierda del multiplicador. Como es 1 se rota el resultado a la derecha y se suma el multiplicando.

		1	1	0	1	0	1	
X			1	0	0	1	1	
	0	0	0	0	0	0	0	0
			1	1	0	1	0	1
			1	1	0	1	0	1

Segundo paso: se toma el siguiente bit de izquierda a derecha del multiplicador. Como es 0 se rota el resultado a la derecha.

		1	1	0	1	0	1	
X			1	0	0	1	1	
	0	0	0	0	0	0	0	
			1	1	0	1	0	1
			1	1	0	1	0	1	0

Tercer paso: se toma el siguiente bit de izquierda a derecha del multiplicador. Como es 0 se rota el resultado a la derecha.

		1	1	0	1	0	1		
X			1	0	0	1	1		
	0	0	0	0	0	0	0		
			1	1	0	1	0	1	
			1	1	0	1	0	1	0	0

Cuarto paso: se toma el siguiente bit de izquierda a derecha del multiplicador. Como es 1 se rota el resultado a la derecha y se suma el multiplicando.

		1	1	0	1	0	1			
X			1	0	0	1	1			
	0	0	0	0	0	0	0			
			1	1	0	1	0	1		
			1	1	0	1	0	1	0	0	0
						1	1	0	1	0	1
			1	1	1	0	1	1	1	0	1

Quinto paso: se toma el siguiente bit de izquierda a derecha del multiplicador. Como es 1 se rota el resultado a la derecha y se suma el multiplicando.

	1	1	0	1	0	1					
X		1	0	0	1	1					
	0	0	0	0	0	0	0				
		1	1	0	1	0	1				
		1	1	0	1	0	1	0	0	0	
					1	1	0	1	0	1	
		1	1	1	0	1	1	1	0	1	0
						1	1	0	1	0	1
		1	1	1	1	1	0	1	1	1	1

El resultado es 1111101111_2

1.9.8 División en binario

El algoritmo de división en binario más comúnmente utilizado se basa en el algoritmo de restas sucesivas. Este algoritmo debe seguir estos pasos:

Se iguala el número de bits en el minuendo y en el divisor agregando ceros a la izquierda en el número con menor cantidad de bits.

Se inicializa el cociente con 0.

Si el dividendo es mayor que el divisor, se le resta al dividendo el divisor y se incrementa el cociente. El resultado de la resta pasa a ser el nuevo dividendo para la siguiente resta.

Si el dividendo es menor que el divisor, la división ha terminado. El resultado está determinado por el cociente y el resultado de la última resta.

Se repite el paso 2 y 3 hasta que el dividendo sea menor que el divisor.

Por ejemplo la división de 11011_2 / 110 es:

Primer paso: como el dividendo es mayor que el divisor, se resta el dividendo menos el divisor. Se incrementa el cociente y el resultado de la resta pasa a ser el nuevo divisor.

11011	00110
11001	1
110100	
+1	
10101	

Segundo paso: como el dividendo es mayor que el divisor, se resta el dividendo menos el divisor. Se incrementa el cociente y el resultado de la resta pasa a ser el nuevo divisor.

```
 11011 | 00110
 11001   10
------
110100
   +1
------
 10101
 11001
------
101110
   +1
------
 01111
```

Tercer paso: como el dividendo es mayor que el divisor, se resta el dividendo menos el divisor. Se incrementa el cociente y el resultado de la resta pasa a ser el nuevo divisor.

```
 11011 | 00110
 11001   11
------
110100
   +1
------
 10101
 11001
------
101110
   +1
------
 01111
 11001
------
101000
   +1
------
 01001
```

Cuarto paso: como el dividendo es mayor que el divisor, se resta el dividendo menos el divisor. Se incrementa el cociente y el resultado de la resta pasa a ser el nuevo divisor.

11011	00110
11001	100
110100	
+1	
10101	
11001	
101110	
+1	
01111	
11001	
101000	
+1	
01001	
11001	
100010	
+1	
00011	

Quinto paso: como el dividendo es menor que el divisor, el resultado de la división es: 100_2 con residuo 11_2.

Ejercicios propuestos

1. Convertir de decimal a binario el número 1845.
2. Convertir de decimal a octal el número 2351.
3. Convertir de decimal a hexadecimal el número 5694.
4. Convertir de binario a decimal el número 10010101101.
5. Convertir de binario a octal el número 10010010100.
6. Convertir de binario a hexadecimal el número 110101100010.
7. Convertir de octal a decimal el número 7461.
8. Convertir de octal a binario el número 6541.
9. Convertir de hexadecimal a decimal el número A45F.
10. Convertir de hexadecimal a binario el número BC20.
11. Sumar en binario los números 101101 + 100110
12. Sumar en octal los números 7461 + 6201
13. Sumar en hexadecimal los números ACD + B49
14. Restar en binario los números 101101 - 100110
15. Restar en octal los números 7461 - 6201
16. Restar en hexadecimal los números ACD - B49
17. Multiplicar en binario los números 100101110 x 100101011
18. Dividir en binario los números 100101110 x 11011

Capítulo 2
Compuertas lógicas

Las compuertas lógicas son elementos de electrónica digital que permiten realizar operaciones lógicas entre cantidades binarias. El álgebra de Boole es el área matemática específica que estudia las operaciones lógicas y sus propiedades.

2.1 COMPUERTA NOT

La compuerta NOT, también conocida como negador o inversor, es una compuerta que permite realizar la operación lógica NOT. Esta operación indica que dada una entrada binaria, su salida será el valor contrario. Es decir, si a la compuerta NOT entra un cero lógico, su salida será un uno lógico y si entra un uno lógico, su salida será un cero lógico.

Cuando se habla de operaciones lógicas, se habla de niveles de verdad. Un cero lógico equivale a falso y un uno lógico equivale a verdadero.

Entonces los resultados de un circuito lógico, están determinados por un concepto denominado "Tabla de verdad", la cual es una tabla que ofrece un valor lógico de salida para una combinación de entrada.

La tabla de verdad de la operación lógica NOT es:

Tabla 2.1 Tabla de verdad de la compuerta NOT

Entrada	Salida
0	1
1	0

La compuerta NOT también cuenta con un símbolo que la hace distinguir de otras compuertas. El símbolo se representa en figura 2.1:

Figura 2.1 Símbolo de la compuerta NOT

Además, en el álgebra de Boole, la compuerta NOT también tiene su expresión. Una entrada o variable, debe especificarse por una letra, igualmente una salida. Entonces si la entrada de la compuerta NOT se denomina A y la salida X, la expresión Booleana de la operación lógica NOT es:

$$X = \bar{A}$$

El funcionamiento de la compuerta se puede observar en la siguiente figura:

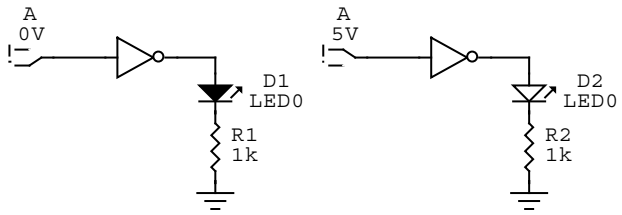


Figura 2.2 Funcionamiento de la compuerta NOT

En la gráfica anterior se puede apreciar que en la figura de la izquierda hay a la entrada un cero lógico por consiguiente a la salida se enciende el LED indicando un uno lógico. En la figura de la derecha hay a la entrada un uno lógico por consiguiente a la salida se apaga el LED indicando un cero lógico.

La referencia de esta compuerta es el número 74LS04, el cual tiene la estructura interna de la figura 2.3.

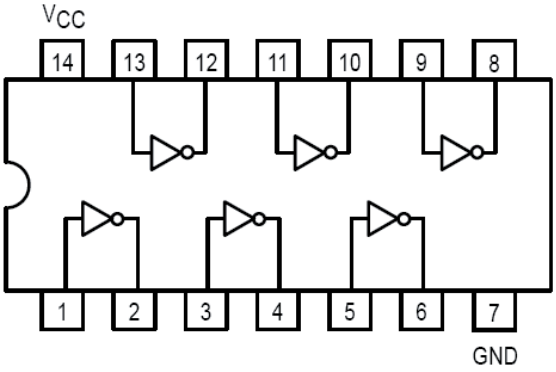


Figura 2.3 74LS04 Compuerta NOT

El circuito 74LS04, tiene internamente 6 compuertas NOT, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 Voltios denominado Vcc

2.2 COMPUERTA AND

La compuerta AND, es una compuerta que permite realizar la operación lógica AND. La operación lógica AND tiene siempre como mínimo dos entradas. La tabla de verdad de la operación lógica AND de dos entradas es:

Tabla 2.2 Tabla de verdad de la compuerta AND de dos entradas

Entradas		**Salida**
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

La tabla de verdad de la operación lógica de tres entradas es:

Tabla 2.3 Tabla de verdad de la compuerta AND de tres entradas

Entradas			Salida
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Con base en las tablas anteriores se puede apreciar una característica fundamental de la operación lógica AND. La salida será uno lógico si y solo si todas sus entradas son uno lógico.

El símbolo de la compuerta AND es:

Figura 2.4 Símbolo de la compuerta AND

En el álgebra de Boole, la expresión de la compuerta AND es:

$$X = A.B$$

El funcionamiento de la compuerta se puede observar en la siguiente figura:

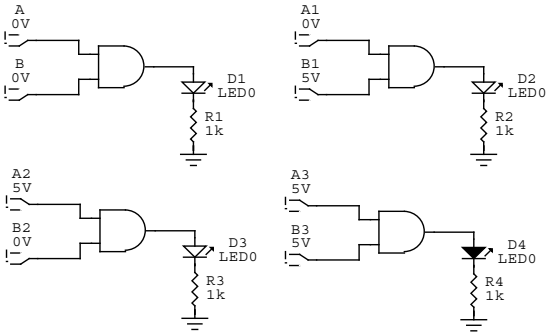


Figura 2.5 Funcionamiento de la compuerta AND

En la figura 2.5 anterior se puede apreciar que en la parte inferior derecha hay en las entradas un uno lógico por consiguiente a la salida se enciende el LED indicando un uno lógico. En el resto de las imágenes la salida es cero lógico, lo cual describe la tabla de verdad de esta operación lógica.

La referencia de la compuerta AND de dos entradas es el número 74LS08, el cual tiene la siguiente estructura interna.

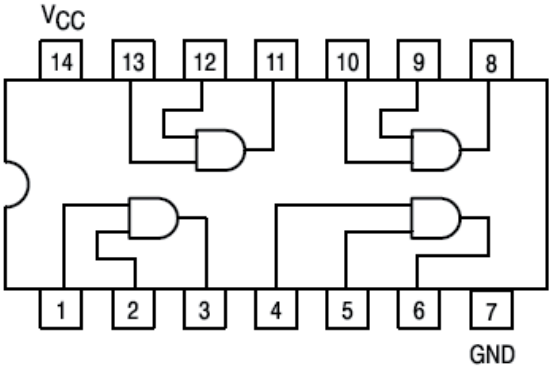


Figura 2.6 74LS08 Compuerta AND de dos entradas

El circuito 74LS08, tiene internamente 4 compuertas AND de dos entradas, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 voltios denominado Vcc.

La referencia de la compuerta AND de tres entradas es el número 74LS11, el cual tiene la siguiente estructura interna.

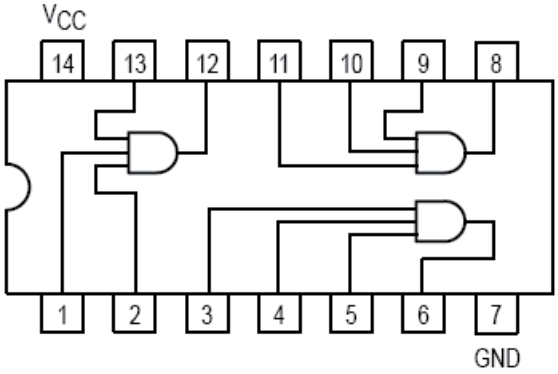


Figura 2.7 74LS11 Compuerta AND de tres entradas

El circuito 74LS11, tiene internamente 3 compuertas AND de tres entradas, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 voltios denominado Vcc.

2.3 COMPUERTA OR

La compuerta OR es una compuerta que permite realizar la operación lógica OR. La operación lógica OR tiene siempre como mínimo dos entradas. La tabla de verdad de la operación lógica OR de dos entradas es:

Tabla 2.4 Tabla de verdad de la compuerta OR de dos entradas

Entradas		Salida
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

La tabla de verdad de la operación lógica de tres entradas es:

Tabla 2.5 Tabla de verdad de la compuerta OR de tres entradas

Entradas			Salida
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Con base en las tablas anteriores se puede apreciar una característica fundamental de la operación lógica OR. La salida será cero lógico sí y sólo sí todas sus entradas son cero lógico.

El símbolo de la compuerta OR se representa en la figura 2.8:

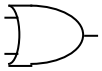


Figura 2.8 Símbolo de la compuerta OR

En el álgebra de Boole, la expresión de la compuerta OR es:

$$X = A + B$$

El funcionamiento de la compuerta se puede observar en la siguiente figura:

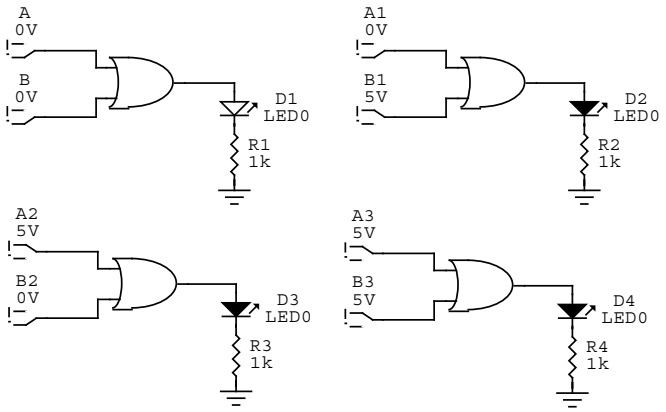


Figura 2.9 Funcionamiento de la compuerta OR

En la figura anterior se puede apreciar que en la parte superior izquierda hay en las entradas ceros lógicos por consiguiente a la salida se apaga el LED indicando un cero lógico. En el resto de las figuras la salida es uno lógico, lo cual describe la tabla de verdad de esta operación lógica.

La referencia de la compuerta OR de dos entradas es el número 74LS32, el cual tiene la siguiente estructura interna.

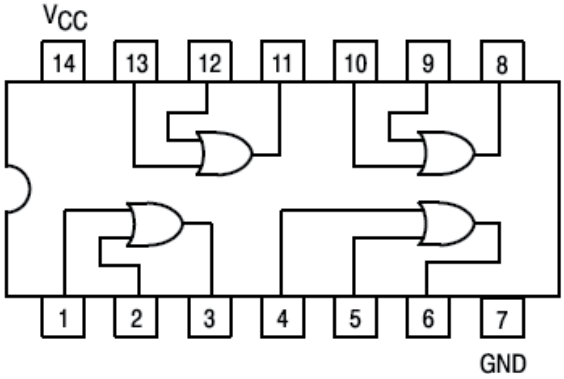


Figura 2.10 74LS32 Compuerta OR de dos entradas

El circuito 74LS32, tiene internamente 4 compuertas OR de dos entradas, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 voltios denominado Vcc.

2.4. COMPUERTA NAND

La compuerta NAND, es una compuerta que permite realizar la operación lógica NAND. La operación lógica NAND tiene siempre como mínimo dos entradas. La tabla de verdad de la operación lógica NAND de dos entradas es:

Tabla 2.6 Tabla de verdad de la compuerta NAND de dos entradas

Entradas		Salida
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

La tabla de verdad de la operación lógica de tres entradas es:

Tabla 2.7 Tabla de verdad de la compuerta NAND de tres entradas

Entradas			Salida
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Con base en las tablas anteriores se puede apreciar una característica fundamental de la operación lógica NAND. La salida será cero lógico sí y solo sí todas sus entradas son uno lógico.

El símbolo de la compuerta NAND es:

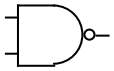


Figura 2.11 Símbolo de la compuerta NAND

En el álgebra de Boole, la expresión de la compuerta AND es:

$$X = \overline{A.B}$$

El funcionamiento de la compuerta se puede observar en la siguiente figura:

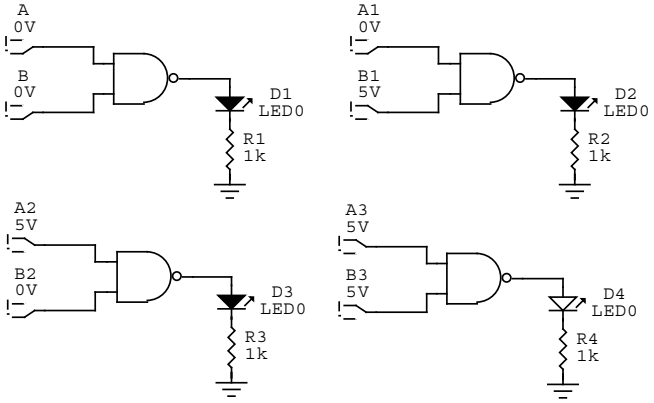


Figura 2.12 Funcionamiento de la compuerta NAND

Se puede apreciar que en la representación de la derecha inferior hay en las entradas uno lógicos por consiguiente a la salida se apaga el LED indicando un cero lógico. En el resto de las figuras la salida es uno lógico, lo cual describe la tabla de verdad de esta operación lógica.

La referencia de la compuerta NAND de dos entradas es el número 74LS00, el cual tiene la siguiente estructura interna.

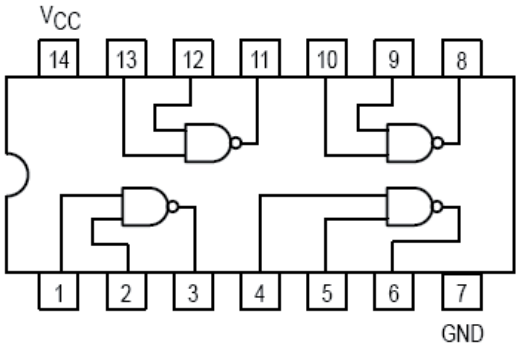


Figura 2.13 74LS00 Compuerta NAND de dos entradas

El circuito 74LS00, tiene internamente 4 compuertas NAND de dos entradas, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 voltios denominado Vcc.

La referencia de la compuerta NAND de tres entradas es el número 74LS10, el cual tiene la siguiente estructura interna.

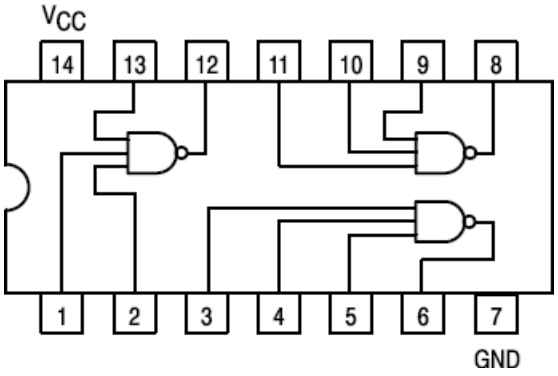


Figura 2.14 74LS10 Compuerta NAND de tres entradas

El circuito 74LS10, tiene internamente 3 compuertas NAND de tres entradas, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 Voltios denominado Vcc.

2.5 COMPUERTA NOR

La compuerta NOR, es una compuerta que permite realizar la operación lógica NOR. La operación lógica NOR tiene siempre como mínimo dos entradas. La tabla de verdad de la operación lógica NOR de dos entradas es:

Tabla 2.8 Tabla de verdad de la compuerta NOR de dos entradas

Entradas		Salida
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

La tabla de verdad de la operación lógica de tres entradas es:

Tabla 2.9 Tabla de verdad de la compuerta NOR de tres entradas

Entradas			Salida
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Con base en las tablas anteriores se puede apreciar una característica fundamental de la operación lógica NOR. La salida será uno lógico si y solo si todas sus entradas son cero lógico.

El símbolo de la compuerta NOR es:

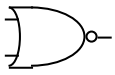


Figura 2.15 Símbolo de la compuerta NOR

En el álgebra de Boole, la expresión de la compuerta NOR es:

$$X = \overline{A + B}$$

El funcionamiento de la compuerta se puede observar en la siguiente figura:

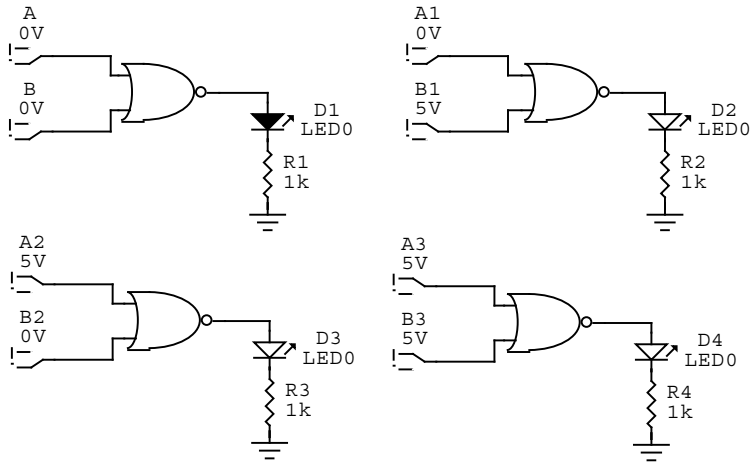


Figura 2.16 Funcionamiento de la compuerta NOR

En la gráfica anterior se puede apreciar que en la figura de la izquierda arriba hay en las entradas ceros lógicos por consiguiente a la salida se enciende el LED indicando un uno lógico. En el resto de las figuras la salida es cero lógico, lo cual describe la tabla de verdad de esta operación lógica.

La referencia de la compuerta NOR de dos entradas es el número 74LS02, el cual tiene la siguiente estructura interna.

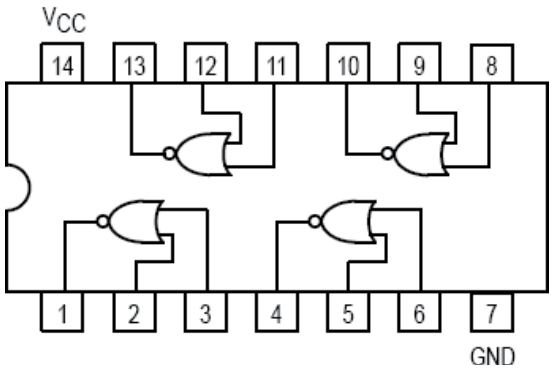


Figura 2.17 74LS02 Compuerta OR de dos entradas

El circuito 74LS02, tiene internamente 4 compuertas NOR de dos entradas, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 voltios denominado Vcc.

La referencia de la compuerta NOR de tres entradas es el número 74LS27, el cual tiene la siguiente estructura interna.

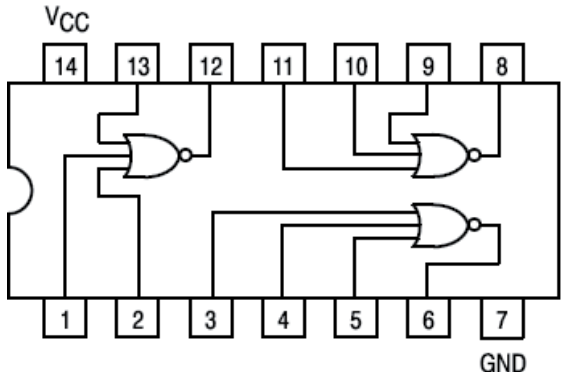


Figura 2.18 74LS27 Compuerta NOR de tres entradas

El circuito 74LS27, tiene internamente 3 compuertas NOR de tres entradas, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 Voltios denominado Vcc.

2.6 COMPUERTA OR EXCLUSIVA

La compuerta OR Exclusiva o también conocida como XOR, es una compuerta que permite realizar la operación lógica XOR. La operación lógica XOR tiene siempre como mínimo dos entradas. La tabla de verdad de la operación lógica XOR de dos entradas es:

Tabla 2.10 Tabla de verdad de la compuerta XOR de dos entradas

Entradas		Salida
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

La tabla de verdad de la operación lógica de tres entradas es:

Tabla 2.11 Tabla de verdad de la compuerta XOR de tres entradas

Entradas			Salida
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Con base en las tablas anteriores se puede apreciar una característica fundamental de la operación lógica XOR. La salida será uno lógico sí y solo sí la suma de unos lógicos a la entrada es impar.

El símbolo de la compuerta XOR es:

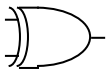


Figura 2.19 Símbolo de la compuerta XOR

En el álgebra de Boole, la expresión de la compuerta XOR es:

$$X \square A \oplus B$$

El funcionamiento de la compuerta se puede observar en la siguiente figura:

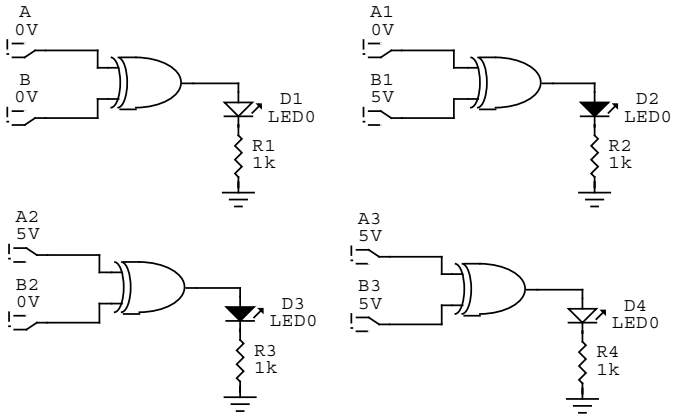


Figura 2.20 Funcionamiento de la compuerta XOR

En la gráfica anterior se puede apreciar que en la primera y última figura hay en las entradas número par de unos lógicos por consiguiente a la salida se apaga el LED indicando un cero lógico. En el resto de las figuras la salida es uno lógico, lo cual describe la tabla de verdad de esta operación lógica.

La referencia de la compuerta XOR de dos entradas es el número 74LS86, el cual tiene la siguiente estructura interna.

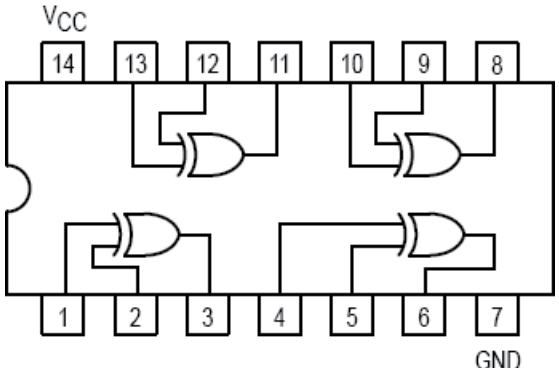


Figura 2.21 74LS86 Compuerta XOR de dos entradas

El circuito 74LS86, tiene internamente 4 compuertas XOR de dos entradas, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 voltios denominado Vcc.

2.7 COMPUERTA NOR EXCLUSIVA

La compuerta NOR Exclusiva o también conocida como XNOR, es una compuerta que permite realizar la operación lógica XNOR. La operación lógica XNOR tiene siempre como mínimo dos entradas. La tabla de verdad de la operación lógica XNOR de dos entradas es:

Tabla 2.12 Tabla de verdad de la compuerta XNOR de dos entradas

Entradas		Salida
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

La tabla de verdad de la operación lógica de tres entradas es:

Tabla 2.13 Tabla de verdad de la compuerta XNOR de tres entradas

Entradas			Salida
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Con base en las tablas anteriores se puede apreciar una característica fundamental de la operación lógica XNOR. La salida será cero lógico sí y solo sí la suma de unos lógicos a la entrada es impar.

El símbolo de la compuerta XNOR es:

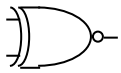


Figura 2.22 Símbolo de la compuerta XNOR

En el álgebra de Boole, la expresión de la compuerta XNOR es:

$$X = \overline{A \oplus B}$$

La implementación de la compuerta XNOR se describe por la siguiente figura:

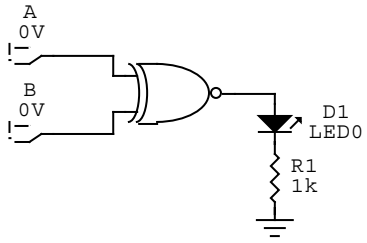


Figura 2.23 Implementación de la compuerta XNOR

El funcionamiento de la compuerta se puede observar en la siguiente figura:

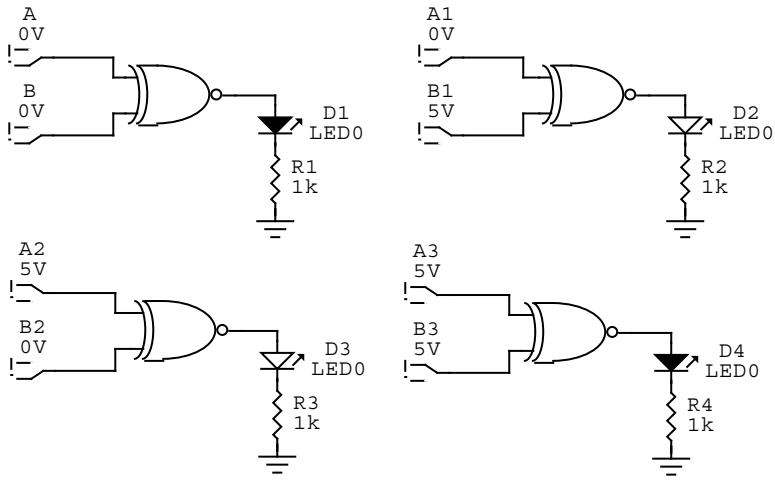


Figura 2.24 Funcionamiento de la compuerta XNOR

En la gráfica anterior se puede apreciar que en la primera y última figura hay en las entradas número par de unos lógicos por consiguiente a la salida se enciende el LED indicando un uno lógico. En el resto de las figuras la salida es cero lógico, lo cual describe la tabla de verdad de esta operación lógica.

La referencia de la compuerta XNOR de dos entradas es el número 74LS266, el cual tiene la siguiente estructura interna.

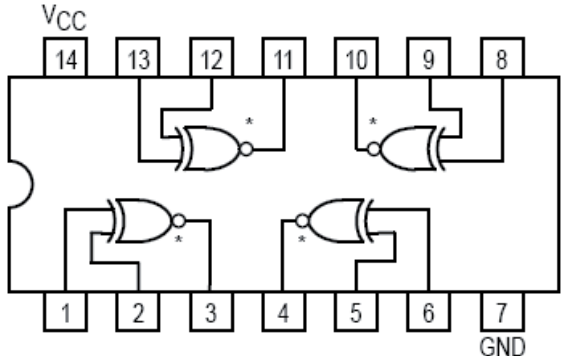


Figura 2.25 74LS266 Compuerta XNOR de dos entradas

El circuito 74LS266, tiene internamente 4 compuertas XNOR de dos entradas de colector abierto, además de un pin que se debe conectar a tierra denominado GND y un pin que se debe conectar a 5 voltios denominado Vcc. La salida de colector abierto indica que la salida debe conectarse a Vcc por medio de una resistencia para que el circuito ejerza el correcto funcionamiento.

2.8. IMPLEMENTACIÓN MEDIANTE COMPUERTAS LÓGICAS

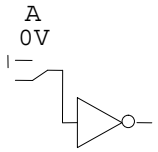
Una expresión lógica puede ser implementada a través de compuertas teniendo en cuenta la prioridad de las operaciones lógicas. La prioridad más alta la determina la operación lógica NOT, es decir que es ésta la primera que se debe implementar. En segundo lugar se encuentra la operación lógica AND y XOR y en último lugar la operación lógica OR, lo que indica que ésta es la última que se debe implementar.

Considerando la siguiente expresión lógica:

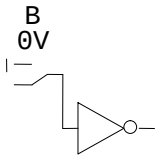
$$X = \overline{A}BC + A\overline{BC} + A \oplus \overline{B}$$

La implementación debe realizarse a través de los siguientes pasos:

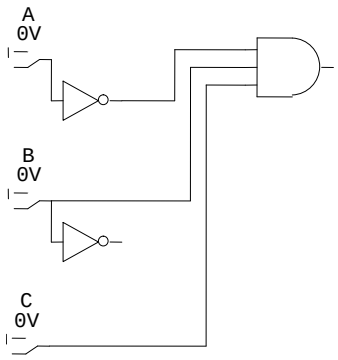
1. Se aplica la operación NOT a la variable A debido a que lo requiere el primer término.



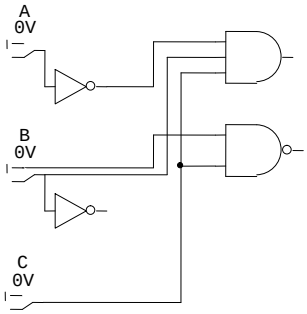
2. Se aplica la operación NOT a la variable B debido a que lo requiere el tercer término.



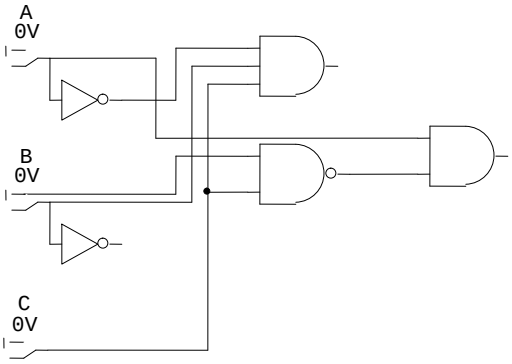
3. Se aplica la operación AND de tres entradas para el primer término, tomando la salida de la operación NOT de la variable A expuesta en el numeral 1, la variable B y la variable C.



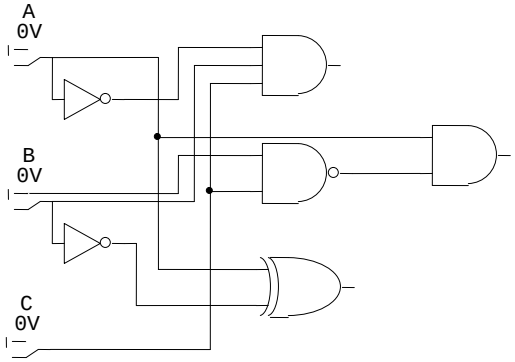
4. Se aplica la operación NAND entre la variable B y C, para usar en el segundo término.



5. Se aplica la operación AND de dos entradas para el segundo término, tomando la salida de la operación NAND expuesta en el numeral 4 y la variable A.



6. Se aplica la operación XOR para el tercer término, tomando la salida de la operación NOT de la variable B expuesta en el numeral 2 y la variable A.



7. Se aplica la operación lógica OR de tres entradas tomando la salida obtenida en la implementación del numeral 3, numeral 5 y numeral 6. Debido a que en la tecnología TTL no existe la compuerta OR de tres entradas, es necesario construirla con base en dos compuertas OR de dos entradas.

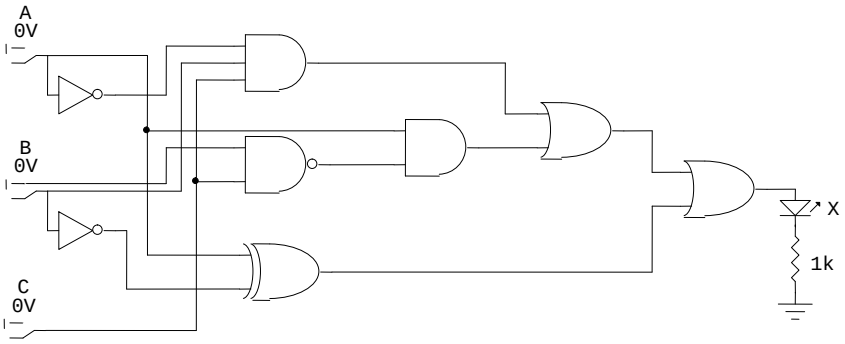


Figura 2.26 Implementación mediante compuertas lógicas

Ejercicios Propuestos

1. Implementar la expresión Booleana $X = \overline{A\overline{B}}C + A\overline{B}C + A\overline{B}$
2. Implementar la expresión Booleana $X = (\overline{A\overline{B}}C \oplus A\overline{B}C) + \overline{(A\overline{B} \oplus AC)}$
3. Implementar la expresión Booleana $X = (A \oplus B)C + AB$
4. Implementar la expresión Booleana $X = ABC + AB\overline{C}$

Capítulo 3

Álgebra de Boole

En 1854 George Boole publicó una obra titulada *Investigación de las leyes del pensamiento, sobre las que se basan las teorías matemáticas de la lógica y la probabilidad*. En esta publicación formuló la idea del álgebra de operaciones lógicas que se conoce hoy en día como Álgebra de Boole. El Álgebra de Boole es una forma que permite expresar y analizar operaciones de circuitos lógicos.

3.1 OPERACIONES BOOLEANAS

En el Álgebra de Boole, una variable es un símbolo que se utiliza para representar magnitudes lógicas. Cualquier variable puede tener un valor de 1 o 0. El complemento es el inverso de la variable. Un literal se define como una variable o su complemento.

3.1.1 Adición Booleana

La adición Booleana es equivalente a la operación lógica OR. En el álgebra de Boole, un término suma es una suma de literales. En circuitos lógicos, un término suma se produce mediante la operación lógica OR de dos literales. El resultado de la suma es 1 cuando uno o más de los literales es uno.

Las reglas de la suma son:

Tabla 3.1 Reglas de la suma Booleana

Entradas		Salida
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

3.1.2 Multiplicación Booleana

La multiplicación Booleana es equivalente a la operación lógica AND. En álgebra de Boole, un término producto es un producto de literales. En circuitos lógicos, un término producto se produce mediante la operación lógica AND de dos literales. El resultado de la multiplicación es 1 cuando todos sus literales son uno.

Las reglas de la multiplicación son:

Tabla 3.2 Reglas de la multiplicación Booleana

Entradas		Salida
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

3.2 LEYES DEL ÁLGEBRA DE BOOLE

Al igual que en otras áreas de las matemáticas, existe en el álgebra de Boole una serie de leyes que resuelven determinados problemas en las implementaciones de circuitos lógicos.

3.2.1 Ley Conmutativa

La ley conmutativa indica que el orden de las variables no altera el resultado. Entonces la ley conmutativa para la adición de dos variables se describe como:

$$A \square B \square B \square A$$

La implementación de la sentencia anterior equivale a:

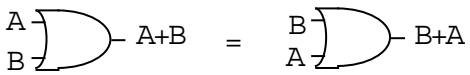


Figura 3.1 Ley conmutativa de la adición Booleana

La ley conmutativa para la multiplicación de dos variables se describe como:

$$A.B \square B.A$$

La implementación de la sentencia anterior equivale a:

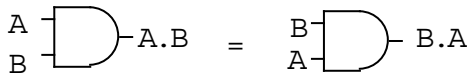


Figura 3.2 Ley conmutativa de la multiplicación Booleana

3.2.2 Ley Asociativa

La ley asociativa para la adición de tres variables se describe como:

$$A \square (B \square C) \square (A \square B) \square C$$

La implementación de la sentencia anterior equivale a:

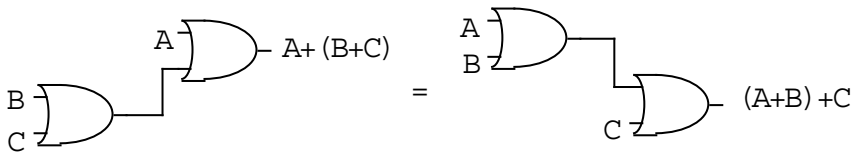


Figura 3.3 Ley asociativa de la adición Booleana

La ley asociativa para la multiplicación de dos variables se describe como:

$$A(B.C) \square (A.B) C$$

La implementación de la sentencia anterior equivale a:

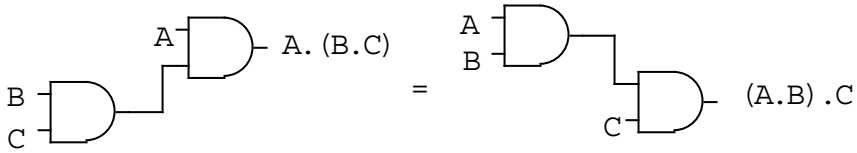


Figura 3.4 Ley asociativa de la multiplicación Booleana

3.2.3 Ley Distributiva

La ley distributiva para tres variables se describe como:

$$A(B \square C) \square A.B \square A.C$$

La implementación de la sentencia anterior equivale a:

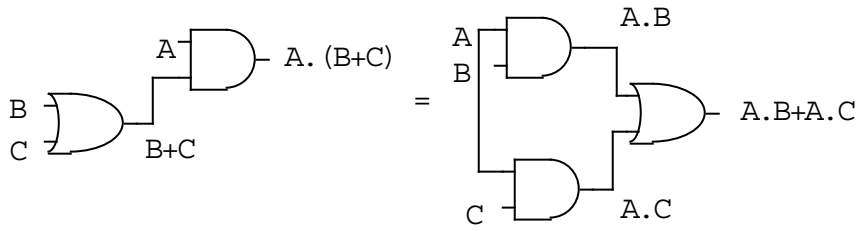


Figura 3.5 Ley distributiva Booleana

3.3 REGLAS DEL ÁLGEBRA DE BOOLE

Existen unas reglas básicas para la manipulación y simplificación de expresiones Booleanas. Estas reglas se describen en la siguiente tabla.

Tabla 3.3 Reglas del álgebra de Boole

Nº	REGLA	DESCRIPCIÓN
1	$A+0=A$	Al aplicar la operación OR a una variable y a cero el resultado es el valor de la variable. $A=0 \Rightarrow 0+0=0$ $A=1 \Rightarrow 1+0=1$
2	$A+1=1$	Al aplicar la operación OR a una variable y a uno el resultado es uno. $A=0 \Rightarrow 0+1=1$ $A=1 \Rightarrow 1+1=1$
3	$A.0=0$	Al aplicar la operación AND a una variable y a cero el resultado es cero. $A=0 \Rightarrow 0.0=0$ $A=1 \Rightarrow 1.0=0$
4	$A.1=A$	Al aplicar la operación AND a una variable y a uno el resultado es el valor de la variable. $A=0 \Rightarrow 0.1=0$ $A=1 \Rightarrow 1.1=1$
5	$A+A=A$	Al aplicar la operación OR a una variable consigo misma el resultado es el valor de la variable. $A=0 \Rightarrow 0+0=0$ $A=1 \Rightarrow 1+1=1$

6	$A+\overline{A}=1$	Al aplicar la operación OR a una variable con su complemento el resultado es 1. $A=0 \Rightarrow 0+1=1$ $A=1 \Rightarrow 1+0=1$
7	$A.A=A$	Al aplicar la operación AND a una variable consigo misma el resultado es el valor de la variable. $A=0 \Rightarrow 0.0=0$ $A=1 \Rightarrow 1.1=1$
8	$A.\overline{A}=0$	Al aplicar la operación AND a una variable con su complemento el resultado es 0. $A=0 \Rightarrow 0.1=0$ $A=1 \Rightarrow 1.0=0$
9	$\overline{\overline{A}}=A$	El doble complemento de una variable es igual a la variable.
10	$A+AB=A$	Aplicando la ley distributiva y las reglas 2 y 4 de esta tabla se puede deducir esta regla. $A+AB=A(1+B)$ $A+AB=A.1$ $A+AB=A$
11	$A+\overline{A}B=A+B$	Aplicando varias reglas de esta tabla se puede deducir esta regla. $A+\overline{A}B=(A+AB)+\overline{A}B$ $A+\overline{A}B=(AA+AB)+\overline{A}B$ $A+\overline{A}B=AA+AB+A\overline{A}+\overline{A}B$ $A+\overline{A}B=(A+\overline{A})(A+B)$ $A+\overline{A}B=1.(A+B)$ $A+\overline{A}B=A+B$
12	$(A+B)(A+C)=A+BC$	Aplicando varias reglas de esta tabla se puede deducir esta regla. $(A+B)(A+C)=AA+AC+AB+BC$ $(A+B)(A+C)=A+AC+AB+BC$ $(A+B)(A+C)=A(1+C+B)+BC$ $(A+B)(A+C)=A.1+BC$ $(A+B)(A+C)=A+BC$

3.4 TEOREMAS DE DEMORGAN

DeMorgan, fue un matemático que estuvo relacionado con Boole. Propuso dos teoremas que están fuertemente relacionados con el álgebra de Boole. Los teoremas de DeMorgan, demuestran las equivalencias entre las compuertas NAND y Negativa OR y las compuertas NOR y Negativa AND.

El primer teorema de DeMorgan se enuncia como: el complemento de un producto de variables es igual a la suma de los complementos de las variables.

$$\overline{XY} = \overline{X} + \overline{Y}$$

Su tabla de verdad es:

Tabla 3.4 Tabla de verdad del primer teorema de DeMorgan

X	Y	$\overline{XY}$	$\overline{X} \square \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

El segundo teorema de DeMorgan se enuncia como: el complemento de una suma de variables es igual al producto de los complementos de las variables.

$$\overline{X \square Y} \square \overline{X}.\overline{Y}$$

Su tabla de verdad es:

Tabla 3.5 Tabla de verdad del segundo teorema de DeMorgan

X	Y	$\overline{X \square Y}$	$\overline{X}.\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Sus compuertas equivalentes son:

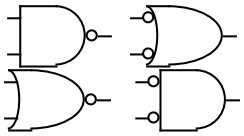


Figura 3.6 Equivalencias del teorema de DeMorgan

3.4.1 Aplicación de los teoremas de DeMorgan

El siguiente procedimiento presenta la aplicación de los teoremas de DeMorgan y el álgebra de Boole a una determinada expresión:

$$X = \overline{\overline{\overline{(A + B\overline{C})} + D.(E + \overline{F})}}$$

Aplicando el segundo teorema de DeMorgan:

$$X = \left(\overline{\overline{(A + B\overline{C})}}\right).\left(\overline{\overline{D.(E + \overline{F})}}\right)$$

Aplicando la regla 9 del álgebra de Boole:

$$X = \left[A + B\overline{C}\right].\left(\overline{\overline{D.(E + \overline{F})}}\right)$$

Aplicando el primer teorema de DeMorgan:

$$X = \left[A + B\overline{C}\right].\left(\overline{D} + \overline{(E + \overline{F})}\right)$$

Aplicando la regla 9 del álgebra de Boole:

$$X = \left[A + B\overline{C}\right].\left[\overline{D} + E + \overline{F}\right]$$

3.5 SIMPLIFICACIÓN MEDIANTE EL ÁLGEBRA DE BOOLE

Cuando se desea implementar alguna expresión con dispositivos lógicos, es muy apropiado reducir la expresión a la forma más simple. Esto asegura el menor costo y esfuerzo, además asegura que la implementación es más eficiente. Mediante las leyes y reglas del álgebra de Boole, además de los teoremas de DeMorgan, se puede reducir una expresión a su forma más simple.

El propósito de la simplificación es usar el menor número posible de compuertas en una implementación.

Por ejemplo, mediante las técnicas del álgebra de Boole, simplificar la siguiente expresión:

$$X = AB + A(B + C) + B(B + C)$$

Aplicando la ley distributiva:

$$X = AB + AB + AC + BB + BC$$

Aplicando la regla 5:

$$X = AB + AC + BB + BC$$

Aplicando la regla 7:

$$X = AB + AC + B + BC$$

Aplicando la regla 10:

$$X = AB + AC + B$$

Aplicando la regla 10:

$$X = AC + B$$

La implementación de las expresiones se presenta en la siguiente figura:

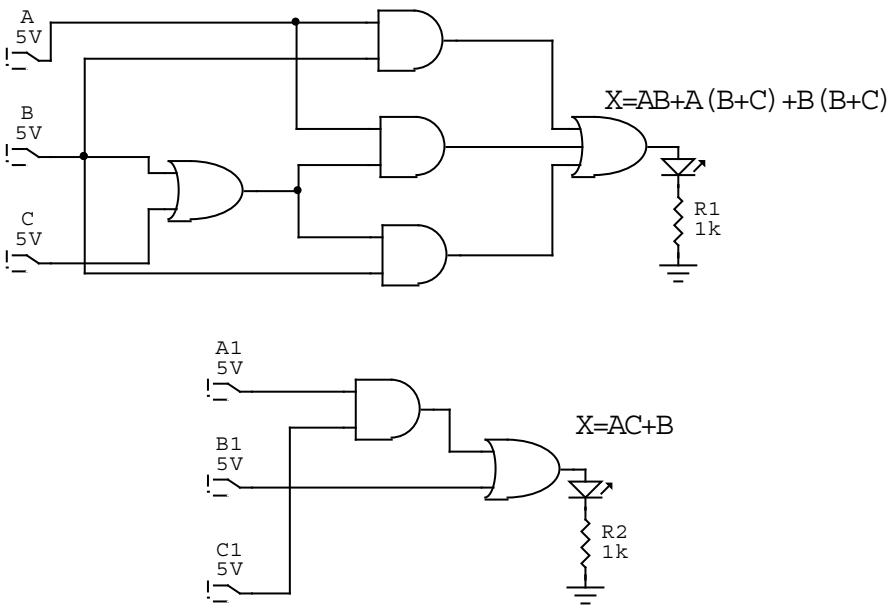


Figura 3.7 Implementación de simplificación mediante álgebra de Boole

3.6 MAPAS DE KARNAUGH

El mapa de Karnaugh proporciona un método para la simplificación de expresiones Booleanas. Si se aplica adecuadamente, genera las expresiones Booleanas más simples posibles.

Un mapa de Karnaugh es similar a una tabla de verdad, ya que muestra los posibles valores de las variables de entrada y de salida. En vez de organizar los valores en filas y columnas, el mapa de Karnaugh lo organiza en celdas, en donde cada celda representa un valor binario de las variables de entrada. El objetivo del mapa de Karnaugh es agrupar celdas adecuadamente. Los mapas de Karnaugh se aplican generalmente para expresiones con 3 o 4 variables. También se puede usar para 2 variables, aunque es muy simple y se puede usar para 5 variables aunque es demasiado complejo. El número de celdas de un mapa de Karnaugh es igual al número total de posibles combinaciones de las variables de entrada. Para 3 variables, el número de celdas es 8. Para 4 variables el número de celdas es 16.

3.6.1 Mapa de Karnaugh de tres variables

El mapa de Karnaugh de 3 variables es una tabla con un número de celdas equivalente a 2^3 celdas donde 3 es el número de variables del mapa. En este caso, A, B y C se utilizan para denominar a las variables. Los valores binarios de A y B se encuentran en la parte izquierda y los valores de C en la parte superior.

AB \ C	0	1
00	$\overline{A}\overline{B}\overline{C}$	$\overline{A}\overline{B}C$
01	$\overline{A}B\overline{C}$	$\overline{A}BC$
11	$AB\overline{C}$	ABC
10	$A\overline{B}\overline{C}$	$A\overline{B}C$

El valor de cada celda corresponde a la combinación especificada en el mapa de Karnaugh anterior. En la primera fila, se puede apreciar que A y B son negados, debido a que el valor de A y B en la parte de la izquierda son 00. Este concepto aplica para todas las celdas del mapa de Karnaugh.

3.6.2 Mapa de Karnaugh de cuatro variables

El mapa de Karnaugh de 4 variables es una tabla con un número de celdas equivalente a 2^4 celdas donde 4 es el número de variables del mapa. En este caso, A, B, C y D se utilizan para denominar a las variables. Los valores binarios de A y B se encuentran en la parte izquierda y los valores de C y D en la parte superior.

AB \ CD	00	01	11	10
00	$\overline{A}\overline{B}\overline{C}\overline{D}$	$\overline{A}\overline{B}\overline{C}D$	$\overline{A}\overline{B}CD$	$\overline{A}\overline{B}C\overline{D}$
01	$\overline{A}B\overline{C}\overline{D}$	$\overline{A}B\overline{C}D$	$\overline{A}BCD$	$\overline{A}BC\overline{D}$
11	$A\overline{B}\overline{C}\overline{D}$	$A\overline{B}\overline{C}D$	$ABCD$	$ABC\overline{D}$
10	$A\overline{B}\overline{C}\overline{D}$	$A\overline{B}\overline{C}D$	$A\overline{B}CD$	$A\overline{B}C\overline{D}$

El valor de cada celda corresponde a la combinación especificada en el mapa de Karnaugh anterior. En la primera fila, se puede apreciar que A y B son negados, debido a que el valor de A y B en la parte de la izquierda son 00. En la primera columna, se puede apreciar que C y D son negados, debido a que el valor de C y D en la parte de la superior son 00. Este concepto aplica para todas las celdas del mapa de Karnaugh.

3.6.3 Minimización de suma de productos mediante un mapa de Karnaugh

El mapa de Karnaugh se utiliza para reducir expresiones Booleanas. Una expresión de suma de productos minimizada está formada por el mínimo número de términos producto posibles con el mínimo número de variables por término. Cada término producto recibe el nombre de "minitérmino". Generalmente una expresión suma de productos minimizada puede ser implementada mediante un número de compuertas menor que su expresión estándar.

Por cada término de la expresión suma de productos, se coloca un 1 en el mapa en la celda correspondiente al valor del producto. Cuando se ha completado el mapa de Karnaugh correspondiente a la suma de productos dada, en dicho mapa deberá haber tantos unos como términos en la expresión. Las celdas que no contienen 1 son aquellas para las que la expresión es 0.

Entonces dada la siguiente expresión,

$$\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + AB\overline{C} + A\overline{B}\overline{C}$$

El mapa de Karnaugh es el siguiente:

AB \ C	0	1
00	1	1
01		
11	1	
10	1	

Dada la siguiente expresión,

$$\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}CD + A\overline{B}\overline{C}D + A\overline{B}CD + ABCD$$

El mapa de Karnaugh es el siguiente:

AB \ CD	00	01	11	10
00	1		1	
01				
11		1	1	
10		1		

3.6.4 Simplificación de suma de productos mediante el mapa de Karnaugh

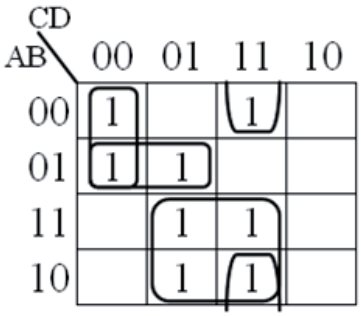
Para simplificar un mapa de Karnaugh con suma de productos, se deben seguir estos pasos:

1. Se colocan los unos en las celdas correspondientes dada la expresión que se desea simplificar.
2. Se agrupan las celdas adyacentes. Una celda adyacente a otra, es aquella en donde sólo cambia una variable. Los grupos se deben hacer de 2, 4, 8 o 16 celdas. Si un mapa, no contiene unos, entonces su resultado es 0. Si en un mapa se agrupan todas las celdas, su resultado es 1. Si un mapa tiene la mitad de celdas en 1 y la mitad de celdas en 0 y no se puede hacer una agrupación, la solución es la XOR o XNOR de las entradas.
3. Cada grupo debe contener el mayor número de celdas posible.
4. Se deben agrupar todos los unos del mapa, sin importar que una celda que contiene 1, se agrupe varias veces.
5. No se deben agrupar celdas que contienen 0.
6. Se analiza cada grupo. Se elimina la variable que en el grupo cambia. Si la variable se conserva, se mantiene su valor para ese minitérmino.

Para revisar el concepto, se muestra los siguientes ejemplos.

Dada la expresión:

$$X = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}CD + A\overline{B}\overline{C}D + A\overline{B}CD + AB\overline{C}D + A\overline{B}\overline{C}D + \overline{A}B\overline{C}D + \overline{A}\overline{B}\overline{C}D$$



Para el primero grupo, se tiene unos en las celdas:

$\overline{A}\overline{B}\overline{C}\overline{D}$ y $\overline{A}B\overline{C}\overline{D}$

Entonces en este grupo se elimina la variable B, debido a que en una celda su valor es 0 y en la otra celda su valor es 1. El minitérmino quedará de la siguiente forma:

$\overline{A}\overline{C}\overline{D}$

Para el segundo grupo, se tiene unos en las celdas:

$\overline{A}B\overline{C}\overline{D}$ y $\overline{A}B\overline{C}D$

Entonces en este grupo se elimina la variable D. El minitérmino quedará de la siguiente forma:

$\overline{A}B\overline{C}$

Para el tercer grupo, se tiene unos en las celdas:

$AB\overline{C}D$, $ABCD$, $A\overline{B}\overline{C}D$ y $A\overline{B}CD$

Entonces en este grupo se elimina las variables B y C. El minitérmino quedará de la siguiente forma:

AB

Para el cuarto grupo, se tiene unos en las celdas:

$\overline{A}\overline{B}CD$ y $A\overline{B}CD$

Entonces en este grupo se elimina la variable A. El minitérmino quedará de la siguiente forma:

$\overline{B}CD$

Finalmente la solución de este mapa de Karnaugh es:

$$X = \overline{A}\overline{C}\overline{D} + \overline{A}B\overline{C} + AD + \overline{B}CD$$

3.6.5 Simplicación de suma de productos usando Tabla de Verdad

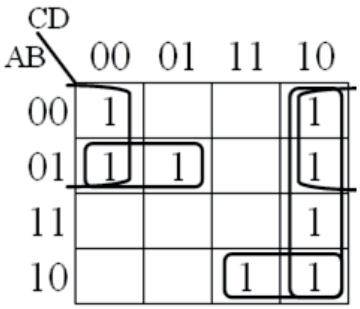
Para simplificar un mapa de Karnaugh con suma de productos, se deben seguir los pasos descritos anteriormente. En esta sección, se solucionará el mapa de Karnaugh tomando como valores de la expresión una tabla de verdad.

Dada la tabla de verdad:

Tabla 3.6 Simplificación mediante mapa de Karnaugh y tabla de verdad

Entradas				Salida
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Se debe colocar los unos de la salida en la celda cuya combinación corresponda. Posteriormente se hacen los grupos posibles.



Finalmente la solución de este mapa de Karnaugh es:

$$X = \overline{A}\,\overline{D} + \overline{A}B\overline{C} + A\overline{B}C + C\overline{D}$$

3.6.6 Condiciones indiferentes o valores No importa

En ocasiones, se tienen códigos de un número de bits pero no hay valor para algunas combinaciones. Por ejemplo para el caso del código BCD, que es un código de cuatro bits, pero sólo se consideran 10 combinaciones. En este caso a las celdas cuyas combinaciones no se consideran, se les coloca un valor "No importa", que se describe con "X". En el momento de hacer los grupos, se puede considerar estos valores como 1 o 0, a conveniencia de la solución del mapa.

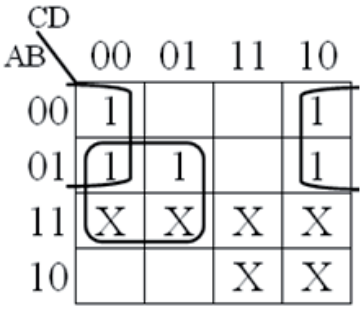
Dada la tabla de verdad:

Tabla 3.7 Simplificación de mapa de Karnaugh con valores No importa

Entradas				Salida
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	X

1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Se debe colocar los unos de la salida y los valores No importa en la celda cuya combinación corresponda. Posteriormente se hacen los grupos posibles.



Finalmente la solución de este mapa de Karnaugh es:

$$X = \overline{A}\overline{D} + B\overline{C}$$

Ejercicios Propuestos

1. Simplificar la expresión Booleana $X = \overline{A\overline{B}C} + A\overline{B}C + A\overline{B}$
2. Simplificar la expresión Booleana $X = (\overline{A\overline{B}C} + A\overline{B}C) + (A\overline{B} + AC)$
3. Simplificar la expresión Booleana $X = ABC + AB\overline{C}$
4. Implementar la expresión Booleana $X = (A \oplus B)C + ABC + (A \oplus B)D$
5. Simplificar la expresión Booleana $\overline{A\overline{B}C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

6. Simplificar mediante mapas de Karnaugh la siguiente tabla de verdad:

Entradas			Salidas	
A	B	C	X	Y
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

7. Simplificar mediante mapas de Karnaugh la siguiente tabla de verdad:

Entradas				Salidas		
A	B	C	D	X	Y	Z
0	0	0	0	0	1	1
0	0	0	1	1	0	1
0	0	1	0	0	1	0
0	0	1	1	0	0	1
0	1	0	0	0	0	0
0	1	0	1	1	1	1
0	1	1	0	0	1	0
0	1	1	1	1	1	1
1	0	0	0	0	0	0

1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	0
1	1	0	0	1	1	1
1	1	0	1	1	1	0
1	1	1	0	0	0	0
1	1	1	1	0	1	0

8. Simplificar mediante mapas de Karnaugh la siguiente tabla de verdad:

Entradas				**Salidas**		
A	B	C	D	X	Y	Z
0	0	0	0	0	1	1
0	0	0	1	1	0	1
0	0	1	0	0	1	1
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	1	1
0	1	1	0	0	1	0
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	1	0	0	0

Capítulo 4

Lógica combinacional

En este capítulo se presentan diferentes tipos de circuitos lógicos combinacionales MSI (*Medium Scale Integration*). Estos dispositivos consisten en circuitos que presentan una salida basado en una entrada combinacional dada.

4.1 SUMADOR BÁSICO

Los sumadores son circuitos muy importantes para diferentes tipos de sistemas digitales en los que se procesan datos numéricos. Para construir un sumador básico es necesario conocer las reglas de la suma binaria. Simplemente se tienen cuatro posibilidades:

0+0=0
0+1=1
1+0=1
1+1=10

Entonces dada las reglas anteriores se puede construir la siguiente tabla de verdad.

Tabla 4.1 Tabla de verdad de sumador básico

A	B	Σ	C_o
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Entonces se obtiene las expresiones Booleanas de las salidas.

$$\sum = \overline{A}B + A\overline{B} = A \oplus B$$

$$C_o = AB$$

Entonces con base en las ecuaciones anteriores se hace la siguiente implementación en la figura 4.1:

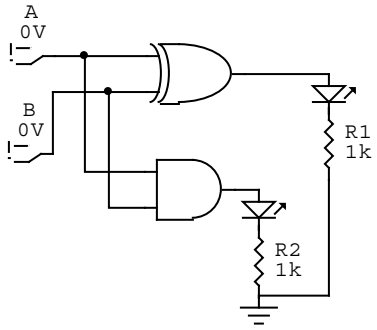


Figura 4.1 Sumador básico

4.2 SUMADOR COMPLETO

El sumador completo se implementa con tres entradas de un bit. Estas tres entradas representan dos entradas de un bit más una tercera que hace referencia a un acarreo de entrada.

Entonces se construye la siguiente tabla de verdad.

Tabla 4.2 Tabla de verdad de sumador completo

A	B	C	Σ	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Entonces se obtiene las expresiones Booleanas de las salidas.

$$\sum = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$$

$$\sum = \overline{A}(\overline{B}C + B\overline{C}) + A(\overline{B}\overline{C} + BC)$$

$$\sum = \overline{A}(B \oplus C) + A(\overline{B \oplus C})$$

$$\sum = A \oplus (B \oplus C)$$

$$C_o = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$C_o = C(\overline{A}B + A\overline{B}) + AB(\overline{C} + C)$$

$$C_o = C(A \oplus B) + AB$$

Entonces con base en las ecuaciones anteriores se hace la siguiente implementación en la figura 4.2:

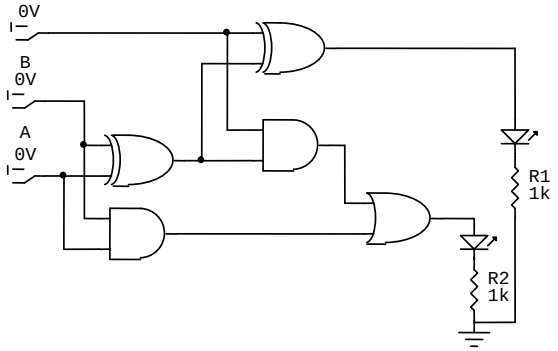


Figura 4.2 Sumador completo

4.3 SUMADOR BINARIO EN PARALELO

Un sumador en paralelo de 4 bits se implementa mediante 4 sumadores completos. Los bits menos significativos, se implementan a la derecha. Los bits de orden más alto, se introducen en el sumador completo que está más a la derecha, aplicando los bits más significativos de cada número al sumador que está más a la izquierda (ver en figura 4.3). La salida de acarreo de cada sumador se conecta a la entrada de acarreo del siguiente sumador de orden superior. Estos se denominan acarreos internos.

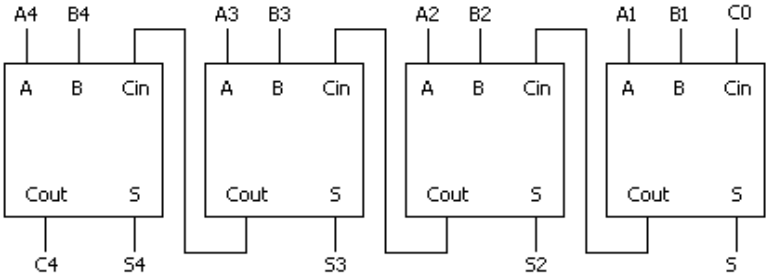


Figura 4.3 Diagrama en bloques de sumador en paralelo de 4 bits

El circuito integrado 74LS283, contiene un sumador en paralelo de 4 bits. La siguiente figura 4.4 muestra la distribución de pines de este circuito.

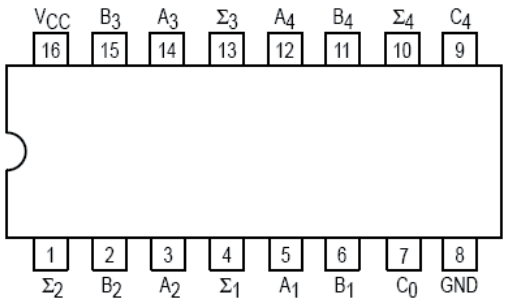


Figura 4.4 74LS283 Sumador en paralelo de 4 bits

4.4 RESTADOR DE 4 BITS

Con base en el sumador completo de 4 bits, es posible aplicar el algoritmo de la resta e implementarlo en un circuito lógico. Para ello, se emplea el algoritmo de la resta visto en el apartado 1.9.4.

Con base en el algoritmo la resta $1101_2 - 1000_2$ es:

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ -\ 1\ 0\ 0\ 0 \\ \hline \end{array}$$

$$\begin{array}{r} 1\ 1\ 1\ 1 \\ 1\ 1\ 0\ 1 \\ +\ 0\ 1\ 1\ 1 \\ \hline 1\ 0\ 1\ 0\ 0 \\ +\ 1 \\ \hline 0\ 1\ 0\ 1 \end{array}$$

El resultado es 0101_2 positivo.

Con base en el algoritmo la resta $1001_2 - 1100_2$ es:

$$
\begin{array}{rcccc}
 & 1 & 0 & 0 & 1 \\
- & 1 & 1 & 0 & 0 \\
\hline
\end{array}
$$

$$
\begin{array}{rcccc}
 & & 1 & 1 & \\
 & 1 & 0 & 0 & 1 \\
+ & 0 & 0 & 1 & 1 \\
\hline
 & 1 & 1 & 0 & 0 \\
\hline
- & 0 & 0 & 1 & 1
\end{array}
$$

El resultado es 0011_2 negativo.

Entonces el algoritmo indica los siguientes pasos:

1. Complementar a 1 el sustraendo. Dada dos entradas A y B, el sustraendo se puede complementar haciendo uso de la compuerta NOT.

2. Sumar el minuendo con el sustraendo complementado. Se puede hacer uso entonces del CI 74283. La siguiente figura muestra estos dos pasos.

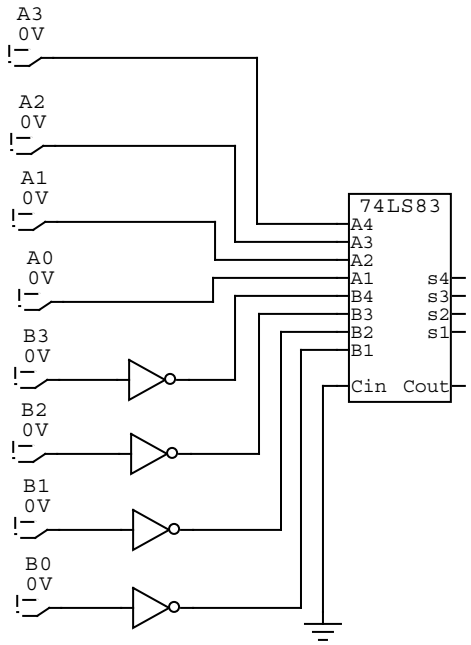


Figura 4.5 Complemento a 1 y suma para la resta binaria

Si el acarreo de salida es 1, entonces se suma al resultado de la suma y el resultado es positivo. Si el acarreo de salida es 0, se debe complementar el resultado de la suma y el resultado es positivo.

Entonces se debe colocar otro sumador completo, en donde la entrada A sea la salida del sumador de la figura anterior. La entrada B con el dato 0000_2 y el acarreo de entrada se conecta al acarreo de salida del primer sumador. Para el caso en que el acarreo de salida del primer sumador mostrado en la figura 4.5, sea 1, se adicionará ese valor al resultado, si es cero se sumará 0, por consiguiente se mantiene el valor.

Si el acarreo es 0, se debe complementar a 1 la salida. Para ello se plantea la siguiente tabla de verdad.

Tabla 4.3 Tabla de verdad para el complemento de la resta binaria

C_o	X_i	Y_i
0	0	1
0	1	0
1	0	0
1	1	1

En donde C_o es el acarreo de salida de la primera suma. X_i es la salida de la segunda suma (esta salida está compuesta por 4 bits) y Y_i es la salida definitiva del restador (esta salida está compuesta por 4 bits). Entonces se obtiene la siguiente expresión Booleana:

$$Y_i = \overline{X_i \oplus C_o}$$

Entonces la implementación definitiva se presenta en la figura 4.6 de la página siguiente.

4.5 COMPARADORES

La función principal de un comparador consiste en comparar dos cantidades binarias para determinar su relación. Es decir, que un circuito comparador determina si dos cantidades son iguales o no.

La compuerta XOR, se puede emplear como un comparador básico ya que su salida es 1 si sus dos bits de entrada son diferentes y la salida es 0 si sus dos bits de entrada son iguales. Si se desea comparar cantidades de varios bit, se requiere una compuerta XOR por cada bit que tenga las cantidades a comparar.

Adicionalmente, por medio de un comparador se puede verificar si una cantidad es mayor que otra usando la compuerta AND. Si se aplica la siguiente expresión Booleana a un circuito:

$$X = A\overline{B}$$

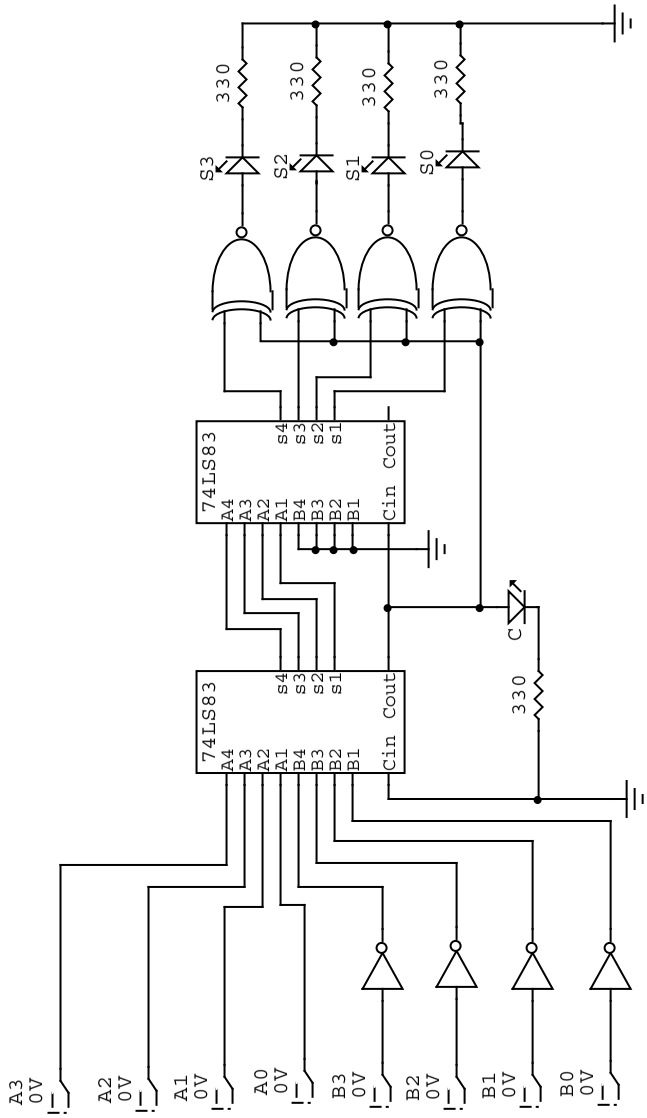


Figura 4.6 Restador de 4 bits

Si A es 1 y B es 0, entonces X es 1 indicando que A>B. Para cualquier otra combinación X es 0 indicando que A<=B

Escalando estos conceptos, se puede implementar un circuito comparador de 2 bits, es decir que tiene dos entradas de 2 bits y es el siguiente en la figura 4.7:

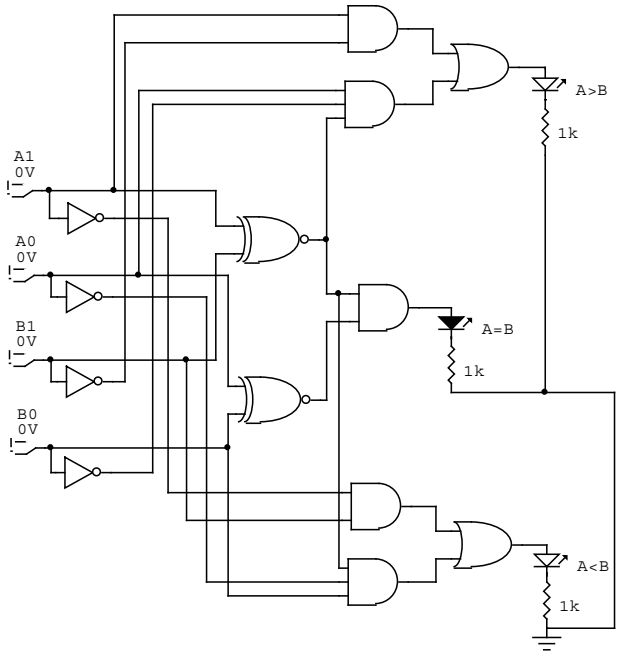


Figura 4.7 Comparador de dos cantidades de dos bits

Existe en el mercado el circuito 74LS85, el cual es un comparador de 4 bits. Además tiene 3 entradas de comparación para conexión en cascada y su representación es la siguiente:

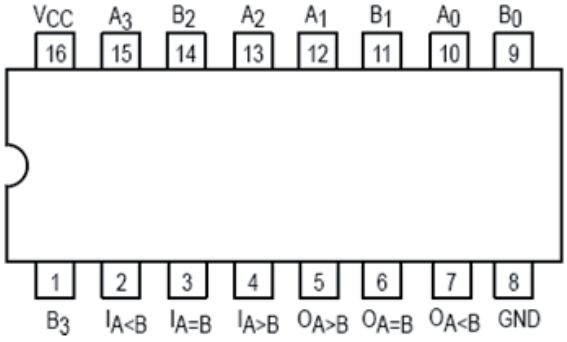


Figura 4.8 74LS85 Comparador de dos cantidades de cuatro bits

4.6 CONVERSORES DE CÓDIGO

Un conversor de código es un circuito combinacional que tiene como entrada un código y como salida otro código.

4.6.1 Conversor Binario-Gray

Para realizar el conversor Binario-Gray es necesario plantear la tabla de verdad.

Tabla 4.4 Tabla de verdad de conversor Binario-Gray

Binario				Gray			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Se elabora un mapa de Karnaugh por cada variable de salida.

G_0

B3B2 \ B1B0	00	01	11	10
00		1		1
01		1		1
11		1		1
10		1		1

$G_0 = \overline{B_1}B_0 + B_1\overline{B_0}$

$G_0 = B_1 \oplus B_0$

G_1

B3B2 \ B1B0	00	01	11	10
00			1	1
01	1	1		
11	1	1		
10			1	1

$G_1 = \overline{B_2}B_1 + B_2\overline{B_1}$

$G_0 = B_2 \oplus B_1$

G_2

B3B2 \ B1B0	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

$G_1 = \overline{B_3} B_2 + B_1 \overline{B_2}$

$G_0 = B_3 \oplus B_2$

G_3

B3B2 \ B1B0	00	01	11	10
00				
01				
11	1	1	1	1
10	1	1	1	1

$G_3 = B_3$

Entonces la implementación es la siguiente:

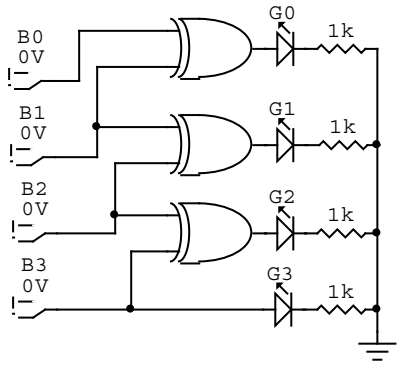


Figura 4.9 Conversor Binario Gray

4.6.2 Conversor Binario-BCD

Para realizar el conversor Binario-BCD es necesario plantear la tabla de verdad.

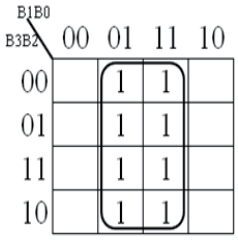
Tabla 4.5 Tabla de verdad de conversor Binario-BCD

Binario				BCD				
B_3	B_2	B_1	B_0	C_4	C_3	C_2	C_1	C_0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1

1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	1
1	1	1	0	1	0	1	0	0
1	1	1	1	1	0	1	0	1

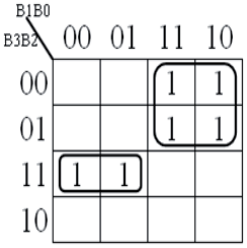
Se elabora un mapa de Karnaugh por cada variable de salida.

C_0



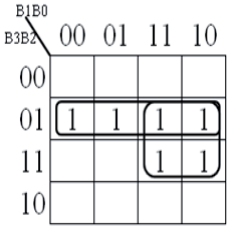
$C_0 = B_0$

C_1



$C_1 = \overline{B_3}B_1 + B_3B_2\overline{B_1}$

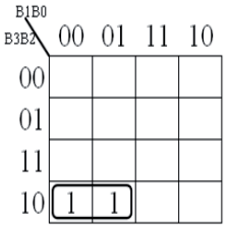
C_2



$C_2 = \overline{B_3}B_2 + B_2B_1$

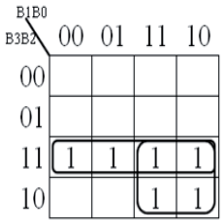
$C_2 = B_2(\overline{B_3} + B_1)$

C_3



$C_3 = B_3\overline{B_2}\,\overline{B_1}$

C_4



$C_4 = B_3B_2 + B_3B_1$

$C_2 = B_3(B_2 + B_1)$

Entonces la implementación es la siguiente:

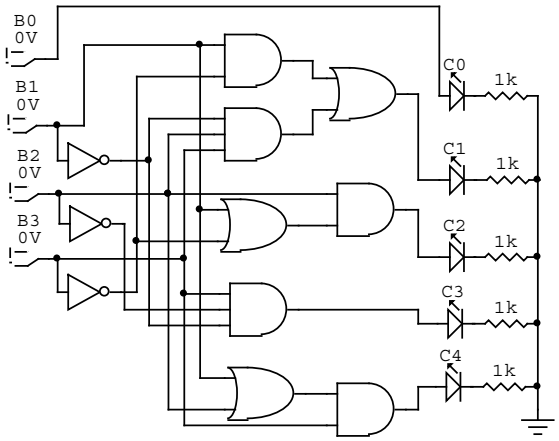


Figura 4.10 Conversor Binario BCD en discreto

Analizando la tabla de verdad, se pueden apreciar dos características importantes. Cuando el valor binario de entrada es menor o igual a 1001_2 la salida en BCD es exactamente igual a la entrada. Sin embargo, cuando la entrada es mayor o igual a 1010_2 en binario, la salida es equivalente al valor en binario de la entrada más 0110_2. Por ejemplo, si la entrada es 1010_2 la salida es 10000_2 lo cual equivale a $1010_2 + 0110_2$. Si la entrada es 1111_2 la salida es 10101_2 lo cual equivale a $1111_2 + 0110_2$. Esto indica que se puede implementar este conversor, sumándole al dato binario 0000_2 si es menor o igual a 1001_2 y sumándole 0110_2 si el dato binario es mayor o igual a 1010_2. La salida C_4 en la implementación anterior precisamente refleja el hecho que el dato binario sea mayor o igual a 1010_2. Entonces se puede plantear la siguiente implementación.

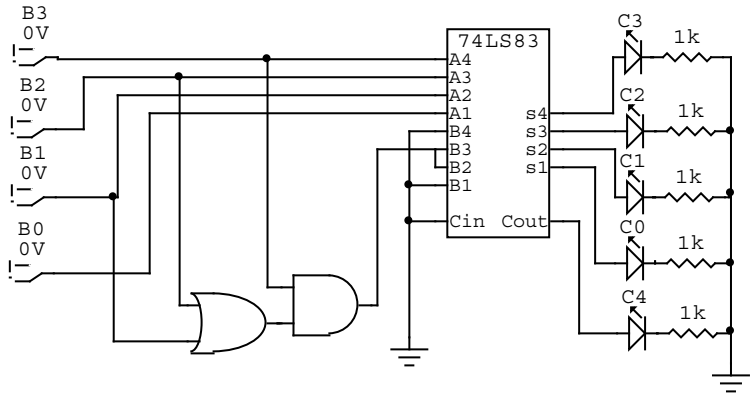


Figura 4.11 Conversor Binario BCD con sumador

4.7 DECODIFICADORES

La función básica del decodificador es detectar la presencia de una determinada combinación de bits en las entradas e indicar la presencia de este código mediante la activación de una línea. Entonces un decodificador posee n líneas de entrada y 2^n líneas a la salida. De esa forma por cada combinación de entrada habrá siempre una sola línea activa a la salida.

4.7.1 Decodificador 2-4

Un decodificador 2-4 indica que tiene dos entradas y cuatro salidas debido a que $2^2 = 4$. Este decodificador tiene la siguiente tabla de verdad.

Tabla 4.6 Tabla de verdad del decodificador 2-4

Binario		Salidas			
B_1	B_0	S_3	S_2	S_1	S_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Entonces las expresiones Booleanas son:

$$S_0 = \overline{B_1}\,\overline{B_0}$$
$$S_1 = \overline{B_1} B_0$$
$$S_2 = B_1 \overline{B_0}$$
$$S_3 = B_1 B_0$$

La implementación de este decodificador es la siguiente:

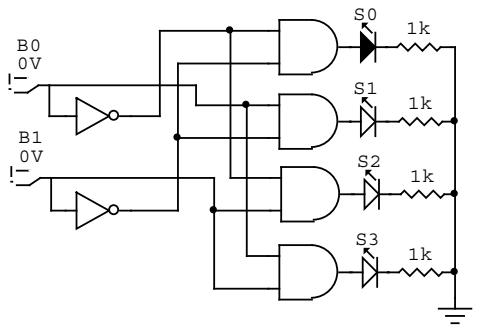


Figura 4.12 Decodificador 2-4

Este decodificador se puede conseguir en el mercado como el CI 74LS139. Este circuito tiene dos decodificadores internamente y sus salidas son activas a nivel bajo. Además tiene una entrada denominada G activa a nivel bajo, que permite habilitar o inhabilitar el circuito. La representación de este circuito se presenta en la siguiente figura 4.13.

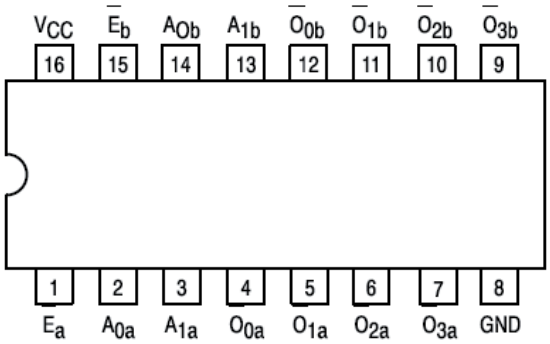


Figura 4.13 74LS139 Decodificador 2-4

4.7.2 Decodificador 3-8

Un decodificador 3-8 indica que tiene tres entradas y ocho salidas debido a que $2^3 = 8$. Este decodificador tiene la siguiente tabla de verdad.

Tabla 4.7 Tabla de verdad del decodificador 3-8

Binario			Salidas							
B_2	B_1	B_0	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Entonces las expresiones Booleanas son:

$$S_0 = \overline{B_2}\,\overline{B_1}\,\overline{B_0}$$

$$S_1 = \overline{B_2}\,\overline{B_1}B_0$$

$$S_2 = \overline{B_2}B_1\overline{B_0}$$

$$S_3 = \overline{B_2} B_1 B_0$$
$$S_4 = B_2 \overline{B_1}\,\overline{B_0}$$
$$S_5 = B_2 \overline{B_1} B_0$$
$$S_6 = B_2 B_1 \overline{B_0}$$
$$S_7 = B_2 B_1 B_0$$

La implementación de este decodificador es la siguiente:

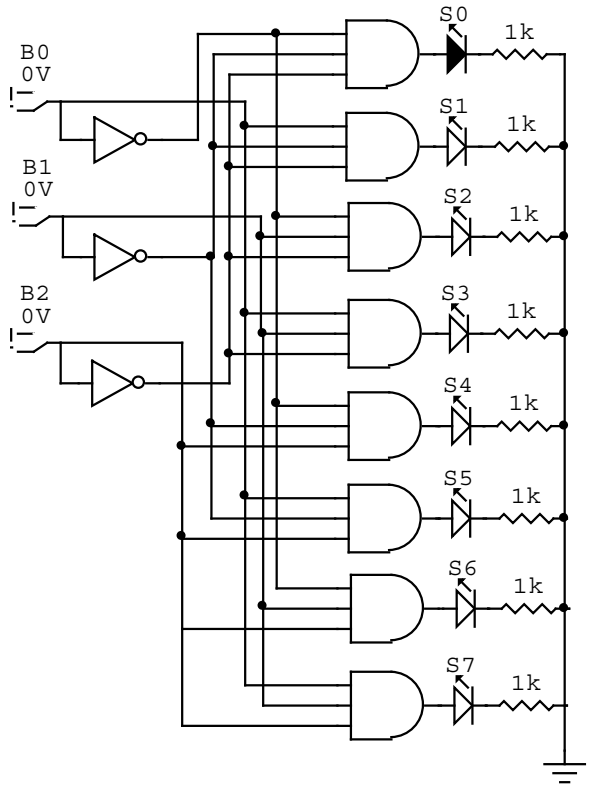


Figura 4.14 Decodificador 3-8

Este decodificador se puede conseguir en el mercado como el CI 74LS138. Este circuito tiene un decodificador y sus salidas son activas a nivel bajo. Además tiene tres entradas denominadas G1, G2A y G2B que permiten habilitar o inhabilitar el circuito. G1 es activa a nivel alto y G2A y G2B son activas a nivel bajo. La representación de este circuito se presenta en la siguiente figura 4.15.

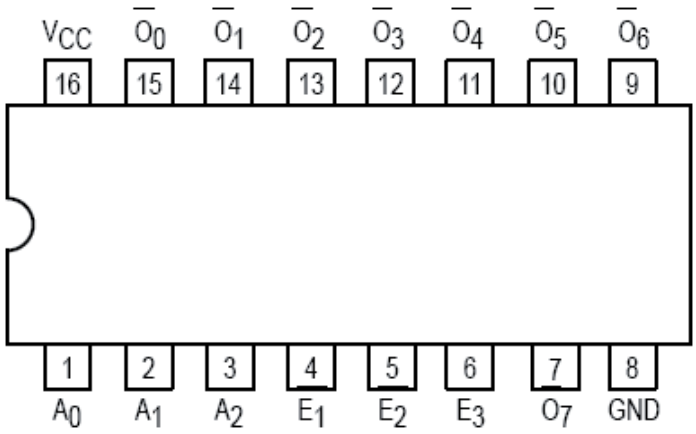


Figura 4.15 74LS138 Decodificador 3-8

4.7.3 Decodificador Manejador

El decodificador manejador es un circuito combinacional que permite recibir un código BCD y generar a sus salidas los niveles lógicos necesarios para encender un display presentando el número decimal determinado por el código BCD.

Un display 7 segmentos es un dispositivo basado en diodos LED, el cual está compuesto por 7 diodos distribuidos de tal forma que al encenderlos se puede formar cualquier dígito decimal. Cada uno de los LEDs están conectados en común por su ánodo o cátodo. Para el caso en que estén conectados por el cátodo, se le denomina cátodo común. Si están conectados por el ánodo, se le denomina ánodo común.

En un display ánodo común, los ánodos de los diodos LED, se encuentran conectados en común están conectados a un pin del display denominado pin común. Los cátodos de los 7 LED, están conectados a un pin exterior del display.

Esto indica que para habilitar un display ánodo común, es necesario colocar su pin común a V_{cc} que equivale a uno lógico. Si se desea encender un segmento se debe colocar GND en el pin correspondiente a ese segmento. Si se desea apagar un segmento se debe colocar V_{cc} en el pin de ese segmento.

Cada segmento en un display se le da un nombre, el cual es una letra minúscula que va de la "a" a la "g" y su distribución es la siguiente:

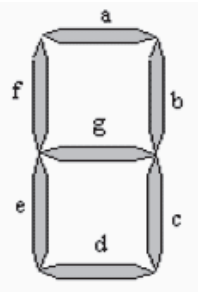


Figura 4.16 Display 7 segmentos

Entonces para diseñar un decodificador manejador se debe realizar su tabla de verdad. Es importante determinar qué tipo de display se usará para el diseño. En este caso el diseño se realizará para un display ánodo común.

Tabla 4.8 Tabla de verdad de un decodificador manejador ánodo común

BCD				Salidas						
B_3	B_2	B_1	B_0	g	f	e	d	c	b	a
0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0	1
0	0	1	0	1	0	1	1	0	1	1
0	0	1	1	1	0	0	1	1	1	1
0	1	0	0	1	1	0	0	1	1	0
0	1	0	1	1	1	0	1	1	0	1
0	1	1	0	0	0	0	0	0	1	1
0	1	1	1	1	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1	0	0	0

Con base en la tabla de verdad 4.8 anterior, se puede obtener las expresiones Booleanas para cada una de las salidas. La implementación permitirá controlar un display 7 segmentos ánodo común.

Existe el CI 74LS47, el cual es un decodificador manejador ánodo común y el CI 74LS48, el cual es un decodificador manejador cátodo común. Su representación es la siguiente:

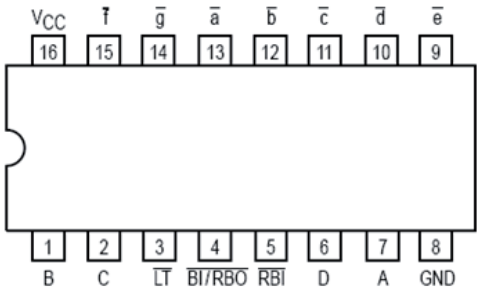


Figura 4.17 74LS47 Decodificador manejador ánodo común

Las entradas A, corresponden al código BCD de entrada. Las salidas a-g corresponden a las salidas que van conectadas al display ánodo común. Estas salidas son activas a nivel bajo ya que con un cero lógico se activa el segmento. La entrada test, sirve para encender todos los segmentos del display. Las líneas RBI y RBO son para hacer conexión en cascada.

La implementación de un decodificador manejador es la siguiente:

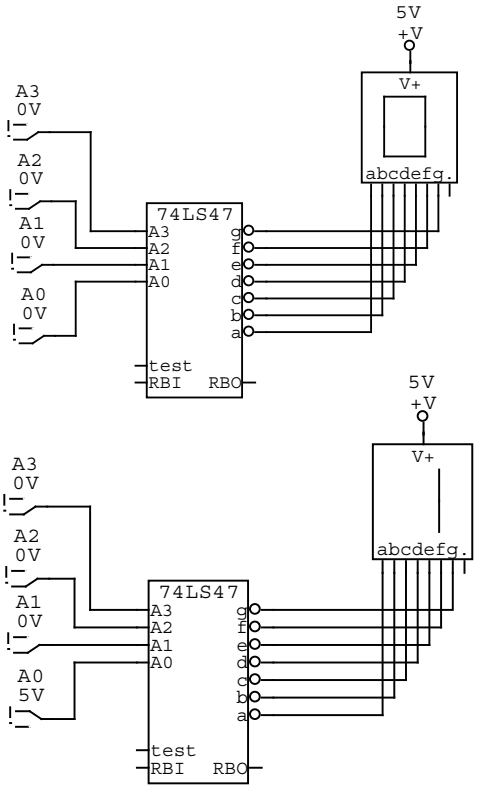


Figura 4.18 Implementación Decodificador Manejador ánodo común

4.8 CODIFICADORES

La función básica del codificador es permitir que se introduzca en alguna de sus entradas un nivel activo y obtener a su salida un código correspondiente a ese nivel.

4.8.1 Codificador Decimal BCD

El uso típico del codificador, es para entradas decimales y salidas BCD. En este caso, hay 10 entradas y 4 salidas. Su tabla de verdad es la siguiente.

Tabla 4.9 Tabla de verdad de un codificador Decimal BCD

Decimal										Salidas			
D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Las expresiones Booleanas basadas de la anterior tabla de verdad son:

$$B_0 = D_1 + D_3 + D_5 + D_7 + D_9$$
$$B_1 = D_2 + D_3 + D_6 + D_7$$
$$B_2 = D_4 + D_5 + D_6 + D_7$$
$$B_3 = D_8 + D_9$$

4.8.2 Codificador Decimal BCD con prioridad

En un circuito donde hay varias entradas, existe la posibilidad de que hallan varias activas a la vez. En estas situaciones, se debe implementar una lógica de prioridad. Para el codificador decimal BCD, se da prioridad a la entrada activa de mayor peso. Entonces la tabla de verdad es la siguiente:

Tabla 4.10 Tabla de verdad de un codificador Decimal BCD con prioridad

Decimal										Salidas			
D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	X	0	**0**	0	1
0	0	0	0	0	0	0	1	X	X	0	0	1	0
0	0	0	0	0	0	1	X	X	X	0	0	1	1
0	0	0	0	0	1	X	X	X	X	0	1	0	0
0	0	0	0	1	X	X	X	X	X	0	1	0	1
0	0	0	1	X	X	X	X	X	X	0	1	1	0
0	0	1	X	X	X	X	X	X	X	0	1	1	1
0	1	X	X	X	X	X	X	X	X	1	0	0	0
1	X	X	X	X	X	X	X	X	X	1	0	0	1

Las expresiones Booleanas basadas de la anterior tabla de verdad son:

$$B_0 = \overline{D_9 D_8 D_7 D_6 D_5 D_4 D_3 D_2 D_1} + \overline{D_9 D_8 D_7 D_6 D_5 D_4 D_3} + \overline{D_9 D_8 D_7 D_6} D_5 + \overline{D_9 D_8} D_7 + D_9$$

$$B_0 = \overline{D_8 D_6 D_4 D_2} D_1 + \overline{D_8 D_6 D_4} D_3 + \overline{D_8 D_6} D_5 + \overline{D_8} D_7 + D_9$$

$$B_1 = \overline{D_9 D_8 D_7 D_6 D_5 D_4 D_3} D_2 + \overline{D_9 D_8 D_7 D_6 D_5 D_4} D_3 + \overline{D_9 D_8 D_7} D_6 + \overline{D_9 D_8} D_7$$

$$B_1 = \overline{D_9 D_8 D_5 D_4} D_2 + \overline{D_9 D_8 D_5 D_4} D_3 + \overline{D_9 D_8} D_6 + \overline{D_9 D_8} D_7$$

$$B_2 = \overline{D_9 D_8 D_7 D_6 D_5} D_4 + \overline{D_9 D_8 D_7 D_6} D_5 + \overline{D_9 D_8 D_7} D_6 + \overline{D_9 D_8} D_7$$

$$B_2 = \overline{D_9 D_8} D_4 + \overline{D_9 D_8} D_5 + \overline{D_9 D_8} D_6 + \overline{D_9 D_8} D_7$$

$$B_3 = \overline{D_9} D_8 + D_9$$

$$B_3 = D_8 + D_9$$

El circuito 74LS147 es un codificador Decimal BCD con prioridad. Este circuito tiene la siguiente representación lógica en figura 4.19:

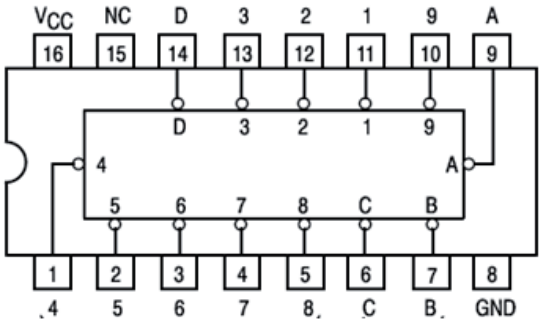


Figura 4.19 74LS147 Codificador Decimal BCD de prioridad.

4.9 DEMULTIPLEXORES

El multiplexor trabaja de forma similar al decodificador. El multiplexor tiene como entrada un código de n bits y como salida 2^n líneas donde habrá una sola activa a la vez. Sin embargo adicionalmente tiene una o varias entradas de información. El objetivo es que la información recibida se presenta sólo en la línea activa.

El CI 74LS139, el cual es un decodificador, se comporta también como demultiplexor, usando sus líneas de habilitación E_a y E_b como entrada de información. De la misma forma trabaja el CI 74LS138, el cual tiene como líneas de información las entradas E_3, E_2 y E_1.

4.10 MULTIPLEXORES

Un multiplexor, también llamado MUX, es un circuito combinacional que permite dirigir la información digital procedente de varias fuentes a una única línea la cual estaría conectada a un destino común.

Los multiplexores tienen n líneas de selección, que permiten seleccionar 2^n entradas y tienen siempre una sola salida.

4.10.1 Multiplexor 4-1

Este multiplexor indica que tiene cuatro entradas de un bit, por consiguiente tiene dos líneas de selección y una sola salida de un bit. Dada la definición, la tabla de verdad para este multiplexor sería la siguiente.

Tabla 4.11 Tabla de verdad de un multiplexor 4-1

Entradas						Salida
S_1	S_0	D_3	D_2	D_1	D_0	Y
0	0	X	X	X	0	0
0	0	X	X	X	1	1
0	1	X	X	0	X	0
0	1	X	X	1	X	1
1	0	X	0	X	X	0
1	0	X	1	X	X	1
1	1	0	X	X	X	0
1	1	1	X	X	X	1

Entonces la expresión Booleana es la siguiente:

$$Y = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$$

La implementación de este circuito es el siguiente:

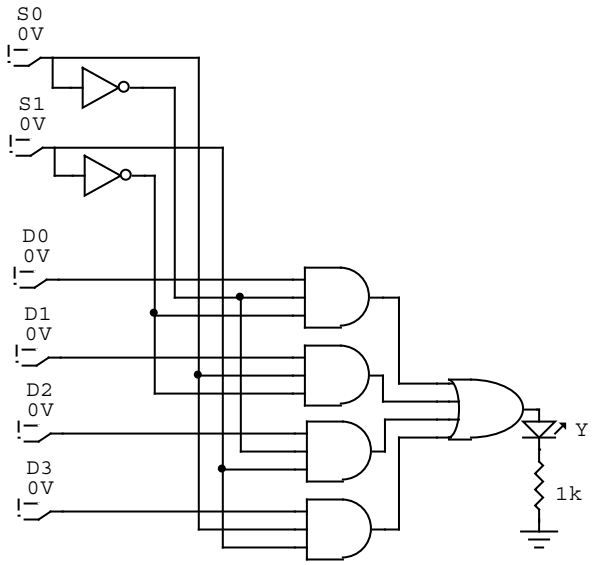


Figura 4.20 Multiplexor 4-1 en discreto

El CI 74LS153 contiene dos multiplexores 4-1. La siguiente figura 4.21 representa este circuito. En este circuito las líneas activas a nivel bajo E_a y E_b son habilitadoras, es decir, que el circuito funcionara si en estas entradas hay un 0 lógico.

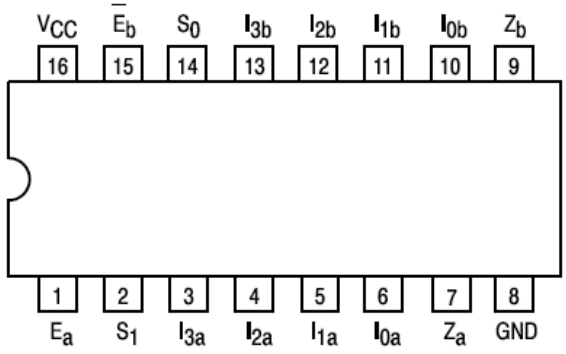


Figura 4.21 74LS153. Multiplexor 4-1

4.10.2 Multiplexor cuádruple 2-1

Este multiplexor indica que tiene dos entradas de cuatro bits, por consiguiente tiene una sola línea de selección y una sola salida de cuatro bits. Dada la definición, la tabla de verdad para este multiplexor sería la siguiente.

Tabla 4.12 Tabla de verdad de un multiplexor 2-4 de 4 bits

Entradas									**Salidas**			
S	A_3	A_2	A_3	A_0	B_3	B_2	B_1	B_0	Y_3	Y_2	Y_1	Y_0
0	a_3	a_2	a_1	a_0	X	X	X	X	a_3	a_2	a_1	a_0
1	X	X	X	X	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0

La tabla anterior indica que si la línea de selección es 0, entonces la salida contendrá lo que tiene la entrada A de 4 bits. Si la línea de selección es 1, entonces la salida contendrá lo que tiene la entrada B de 4 bits.

Las expresiones Booleanas de este circuito son:

$$Y_0 = \overline{S}A_0 + SB_0$$
$$Y_1 = \overline{S}A_1 + SB_1$$
$$Y_2 = \overline{S}A_2 + SB_2$$
$$Y_3 = \overline{S}A_3 + SB_3$$

La implementación de este circuito es la siguiente:

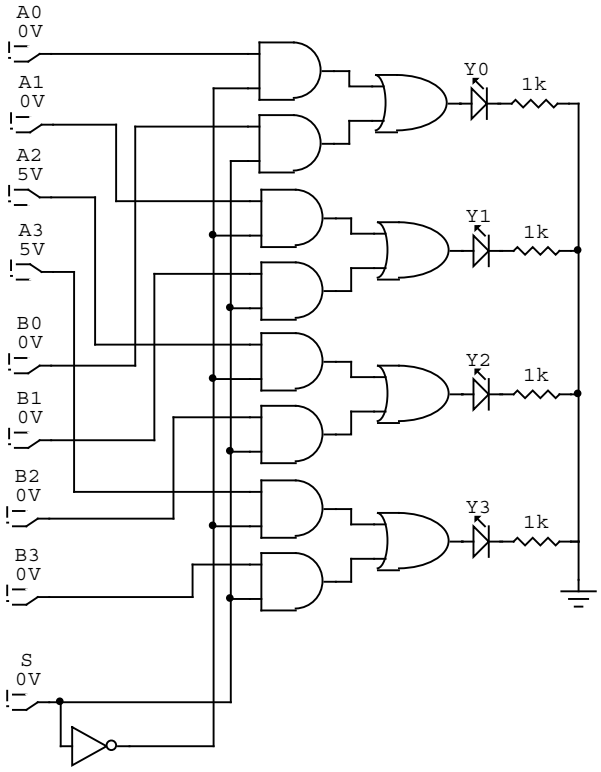


Figura 4.22 Multiplexor 2-1 de 4 bits en discreto

El CI 74LS157 contiene un multiplexor 2-1 de 4 bits. La siguiente figura 4.23 representa este circuito. En este circuito la línea activa a nivel bajo E es un habilitador, es decir, que el circuito funcionará si en esta entrada hay un 0 lógico.

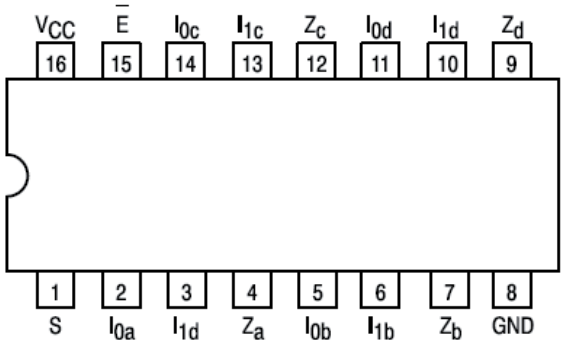


Figura 4.23 74LS157. Multiplexor 2-1 de 4 bits

Ejercicios Propuestos

1. Diseñar e implementar un sumador de 2 cantidades de 8 bits basado en sumadores de 4 bits.

2. Diseñar e implementar un comparador de código binario de 3 bits.

3. Diseñar e implementar un conversor de código Gray a código Binario.

4. Diseñar e implementar un decodificador manejador para manejar un display ánodo común.

5. Diseñar e implementar un decodificador manejador para manejar un display cátodo común.

6. Diseñar e implementar un multiplexor de 4 a 1 de 4 bits.

Capítulo 5

Elementos básicos de almacenamiento

Existen circuitos que permiten almacenar informaciones lógicas basadas en circuitos básicos como compuertas y algunos circuitos de la lógica combinacional. En este capítulo se presentan circuitos biestables, monoestables y aestables. Los circuitos biestables son aquellos que poseen estados de SET (activado) y RESET (desactivado) en los cuales se permite mantener información de forma indefinida. Un circuito monoestable, tiene un único estado estable el cual se produce mediante una activación o disparo que se le proporciona al circuito y su salida tiene un ancho de pulso determinado previamente configurado. El circuito aestable, no posee estado estable lo cual indica que se comporta como un oscilador.

5.1 LATCHES

El latch, es un dispositivo de almacenamiento de dos estados. Este dispositivo, posee realimentación. Dicha realimentación consiste en que una salida se conecta con una estrada opuesta. Este aspecto permite mantener un estado a la salida del circuito.

5.1.1 Latch S-R

El latch S-R es un tipo de multivibrador biestable. Las letras S-R significan Set-Reset con entrada activa a nivel alto. Se compone de dos compuertas NOR acopladas como se muestra en la siguiente figura.

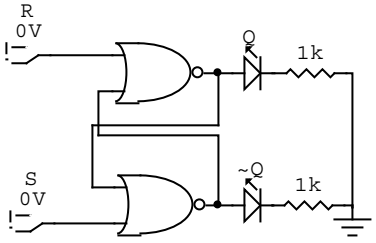


Figura 5.1 Latch S-R

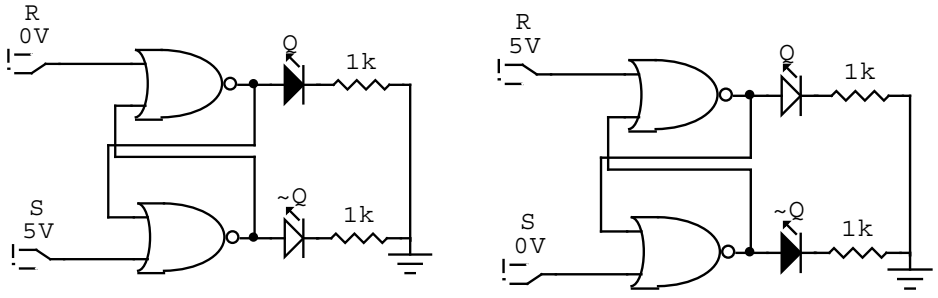
El funcionamiento de este circuito se puede analizar, verificando cada uno de los posibles estados de la entrada del circuito.

Un primer caso, es colocando un 1 lógico en la entrada S y un 0 lógico en la entrada R. Entonces se puede notar que la salida de la segunda compuerta será 0, debido a que una de sus entradas es: 1. La primer compuerta tiene a la entrada R un 0 y de realimentación tiene un 0 entonces la salida es 1. Este caso se denomina Set, que indica que la entrada S tiene un 1, la entrada R tiene un 0 y las salidas quedan Q=1 y ~Q=0.

Un segundo caso, es colocando un 1 lógico en la entrada R y un 0 lógico en la entrada S. Entonces se puede notar que la salida de la primera compuerta será 0, debido a que una de sus entradas es 1. La segunda compuerta tiene a la entrada S un 0 y de realimentación tiene un 0 entonces la salida es 1. Este caso se denomina Reset, que indica que la entrada R tiene un 1, la entrada S tiene un 0 y las salidas quedan Q=0 y ~Q=1.

Un tercer caso, es colocando en las entradas R y S 0 lógico. Para este caso, las salidas de las compuertas NOR, dependen de las entradas de la realimentación. Por consiguiente es necesario considerar condiciones iniciales en la salida. Si se considera que la salida Q=0 y ~Q=1, entonces la salida de la primera compuerta será 0 y la salida de la segunda compuerta será 1. Si se considera que la salida Q=1 y ~Q=0, entonces la salida de la primera compuerta será 1 y la salida de la segunda compuerta será 0. Por consiguiente se puede observar que si la entrada es R=0 y S=0, las salidas conservan su estado actual. A este caso, se le denomina No Cambio.

Un cuarto caso, es colocando en las entradas R y S, 1 lógico. Para este caso, las salidas de las compuertas NOR, siempre serán 0. Al ser este circuito un biestable, se indica que la salida ~Q es el complemento de Q, entonces si ambas salidas son 0, no se cumple esta condición, lo cual indica que este es un caso inválido.



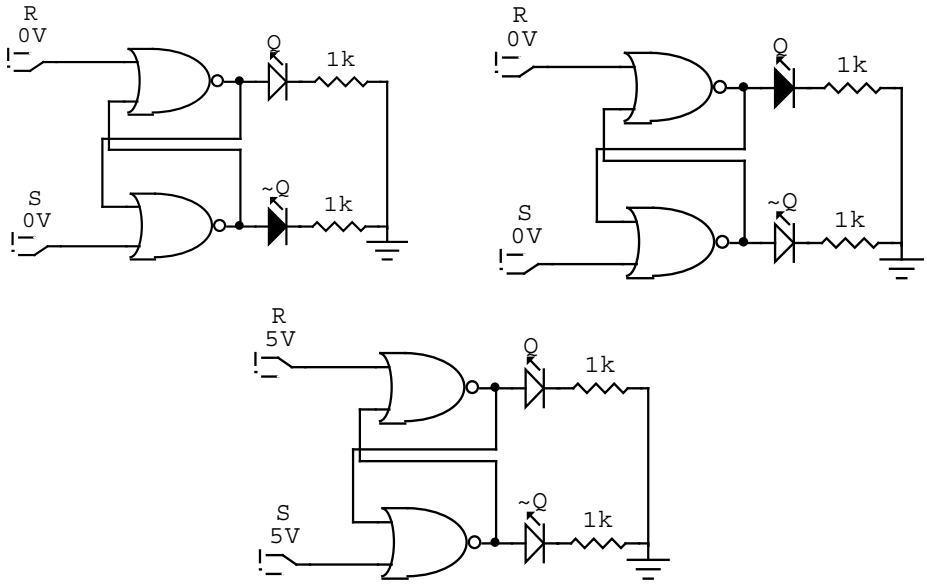


Figura 5.2 Funcionamiento de Latch S-R

Entonces la tabla de verdad del latch S-R es la siguiente:

Tabla 5.1 Tabla de verdad de Latch S-R

Entradas		Salidas		
R	S	Q	~Q	
0	0	NC	NC	No cambio
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	No válido

5.1.2 Latch $\overline{S}-\overline{R}$

Este latch tiene las mismas características que el latch S-R considerando que sus entradas son activas a nivel bajo. Se compone de dos compuertas NAND acopladas como se muestra en la siguiente figura.

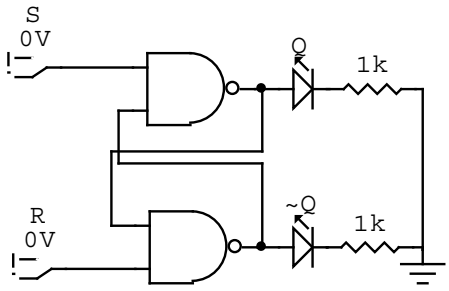


Figura 5.3 Latch $\overline{S} - \overline{R}$

El funcionamiento de este circuito se puede analizar, verificando cada uno de los posibles estados de la entrada del circuito.

Un primer caso, es colocando un 1 lógico en la entrada S y un 0 lógico en la entrada R. Entonces se puede notar que la salida de la segunda compuerta será 1, debido a que una de sus entradas es 0. La primer compuerta tiene a la entrada R un 1 y de realimentación tiene un 1 entonces la salida es 0. Este caso se denomina ***Reset***, que indica que la entrada S tiene un 1, la entrada R tiene un 0 y las salidas quedan Q=0 y ~Q=1.

Un segundo caso, es colocando un 1 lógico en la entrada R y un 0 lógico en la entrada S. Entonces se puede notar que la salida de la primera compuerta será 1, debido a que una de sus entradas es 0. La segunda compuerta tiene a la entrada R un 1 y de realimentación tiene un 1 entonces la salida es 0. Este caso se denomina ***Set***, que indica que la entrada R tiene un 1, la entrada S tiene un 0 y las salidas quedan Q=1 y ~Q=0.

Un tercer caso, es colocando en las entradas R y S, 1 lógico. Para este caso, las salidas de las compuertas NAND, dependen de las entradas de la realimentación. Por consiguiente es necesario considerar condiciones iniciales en la salida. Si se considera que la salida Q=0 y ~Q=1, entonces la salida de la primera compuerta será 0 y la salida de la segunda compuerta será 1. Si se considera que la salida Q=1 y ~Q=0, entonces la salida de la primera compuerta será 1 y la salida de la segunda compuerta será 0. Por consiguiente se puede observar que si la entrada es R=0 y S=0, las salidas conservan su estado actual. A este caso, se le denomina No Cambio.

Un cuarto caso, es colocando en las entradas R y S, 0 lógico. Para este caso, las salidas de las compuertas NAND, siempre serán 1. Al ser este circuito un biestable, se indica que la salida ~Q es el complemento de Q, entonces si ambas salidas son 1, no se cumple esta condición, lo cual indica que este es un caso inválido.

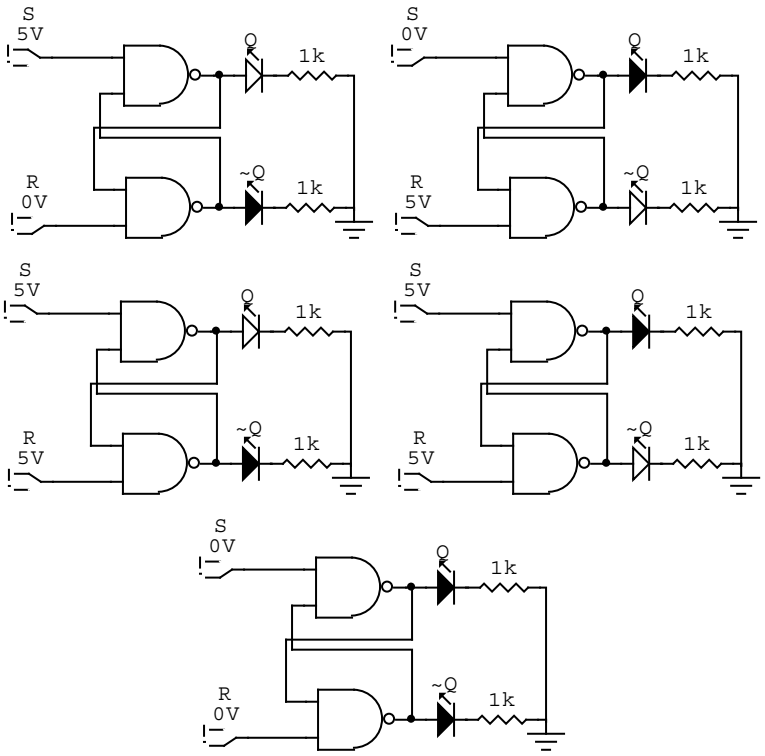


Figura 5.4 Funcionamiento de Latch $\overline{S}-\overline{R}$

Entonces la tabla de verdad del latch $\overline{S}-\overline{R}$ es la siguiente:

Tabla 5.2 Tabla de verdad de Latch $\overline{S}-\overline{R}$

Entradas		Salidas		
~R	~S	Q	~Q	
0	0	1	1	No válido
0	1	0	1	Reset
1	0	1	0	Set
1	1	NC	NC	No cambio

5.1.3 Circuito Antirrebote

En la naturaleza existe siempre un fenómeno físico que describe la ley de Newton de acción y reacción.

Cualquier interruptor al ser posicionado en un estado, genera rebotes. Estos rebotes, generalmente no son visibles, dado que su frecuencia es muy alta. Sin embargo si se conectara un dispositivo a un interruptor, el rebote que genera pequeños pulsos indeseables que un circuito puede detectar. Estos pulsos podrían alterar la respuesta del circuito. Entonces la solución es construir un circuito que elimine el efecto del rebote en los interruptores. Existen varias opciones de implementar circuitos antirrebotes, la siguiente figura muestra un ejemplo.

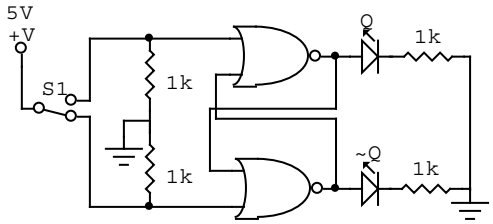


Figura 5.5 Circuito Antirrebote

Para construir este circuito, se requiere un latch S-R y un interruptor de 3 pines, 2 posiciones. Cuando el interruptor coloca un 1 lógico en la entrada R, se genera un ***Reset***, debido a que la entrada S estaría en 0 lógico por medio de la resistencia a tierra. En el momento de haber un rebote, la entrada R y la entrada S, quedan en 0 lógico generando a la salida un estado de no cambio. Una vez finalizado el rebote, vuelve a estar en estado de ***Reset*** hasta quedar la entrada en estado estable. Entonces para el rebote generado en el interruptor de entrada, a las salidas Q y ~Q se obtiene una señal sin rebotes.

5.1.4 Latch S-R con Habilitación

El latch S-R, actúa en el mismo instante en que se modifique las entradas. Si se quisiera pasar de ***Set*** a ***Reset***, sería necesario pasar por el estado No cambio para no alterar las salidas. Sin embargo en este latch se puede cometer el error de pasar por el estado inválido, lo que podría generar un error en la salida del circuito.

La solución al problema planteado es agregarle una línea de habilitación al latch, de tal manera que sólo trabaje cuando esta línea se encuentre activa. Este circuito se muestra en la siguiente figura.

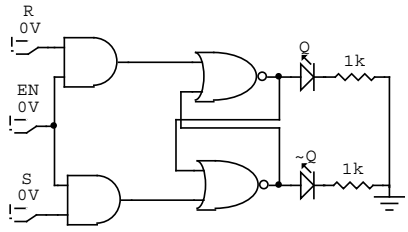


Figura 5.6 Latch S-R con Habilitación

Este circuito trabaja exactamente igual al latch S-R con una función adicional. Si la entrada EN es 0, la salida siempre será No cambio. Este comportamiento describe el almacenamiento de un bit de información. La tabla de verdad es la siguiente:

Tabla 5.3 Tabla de verdad de Latch S-R con Habilitación

Entradas			Salidas		
EN	~R	~S	Q	~Q	
1	0	0	NC	NC	No cambio
1	0	1	1	0	Set
1	1	0	0	1	Reset
1	1	1	0	0	No válido
0	X	X	NC	NC	No cambio

5.1.5 Latch D con Habilitación

El latch S-R con habilitación, sigue teniendo el inconveniente de presentar el estado no válido. El latch D, es una implementación que permite eliminar este estado mejorando el comportamiento del latch. Este circuito se muestra en la siguiente figura.

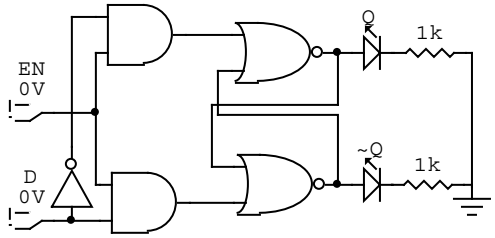


Figura 5.7 Latch D

Este circuito trabaja ahora solo tiene los estados de No cambio, ***set*** y ***reset***. La tabla de verdad es la siguiente:

Tabla 5.4 Tabla de verdad de Latch D

Entradas		Salidas		
EN	D	Q	~Q	
1	0	0	1	Reset
1	1	1	0	Set
0	X	NC	NC	No cambio

5.2 FLIP-FLOPS

Los Flip-Flops son circuitos biestables síncronos. El término síncrono indica que la salida del mismo varía únicamente en un instante de tiempo específico de una entrada denominada reloj. Ese instante de tiempo se denomina ***flanco*** y puede haber flanco de subida y de bajada. El flanco de subida hace referencia al instante de tiempo o impulso existente cuando la entrada de reloj varía del estado 0 a 1. El flanco de bajada hace referencia al instante de tiempo o impulso existente cuando la entrada de reloj varía del estado 1 a 0. El tiempo de duración de un flanco es de aproximadamente 30 nanosegundos.

El Flip-Flop cambia de estado en la salida solamente cuando se presenta un flanco en su entrada de reloj. Los Flip-Flops pueden ser de flanco de subida o de flanco de bajada. Para generar un flanco es necesario implementar un circuito detector de flancos.

5.2.1 Detector de flancos

Para implementar un circuito detector de flancos, es necesario hacer uso de una característica de los circuitos TTL. Toda compuerta, tiene un tiempo de propagación que es aproximadamente 10 ns (este tiempo puede variar según la tecnología de la compuerta). Con base en este concepto se puede obtener un flanco con el circuito de la siguiente figura.

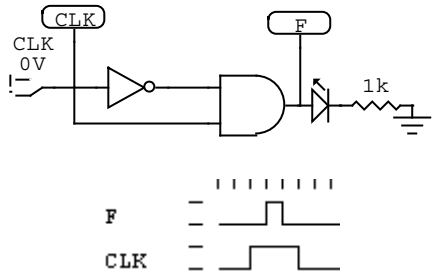


Figura 5.8 Detector de Flancos

Para el circuito anterior, en el instante de tiempo en que la entrada CLK, pasa de 0 a 1, la salida de la compuerta NOT se mantiene en 1 durante 10 ns, permitiendo que la salida de la compuerta AND tenga a sus entradas 1 lógico durante 10 ns, dejando en la salida F un 1 lógico durante 10 ns, el cual se llamará flanco de subida. En el momento en que la entrada CLK, pasa de 1 a 0, a la salida de la compuerta NOT se mantendrá un 0 lógico y antes de que pase a 1 lógico, la otra entrada de la compuerta AND será 0 lógico, por consiguiente no

hará ningún efecto al pasar de 1 a 0 la entrada CLK. Entonces de esta forma se produce un flanco de subida.

El símbolo de un circuito con reloj, puede ser con flanco de subida o flanco de bajada. En otra notación, el reloj se puede identificar con las letras CP (Clock Pulse). La siguiente figura muestra estos símbolos respectivamente.

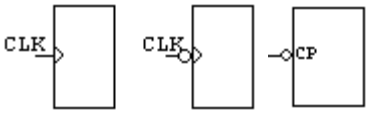


Figura 5.9 Simbología de circuito con entrada de reloj

5.2.2 Flip-Flop D

El Flip-Flop D, es exactamente igual al latch D, reemplazando la entrada de habilitación por un reloj. Para esto simplemente es necesario agregar al latch D, un circuito detector de flancos. La implementación es la siguiente.

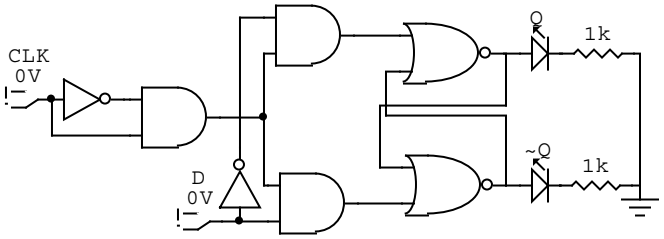


Figura 5.10 Flip-Flop D

La tabla de verdad del Flip-Flop D es:

Tabla 5.5 Tabla de verdad de Flip-Flop D

Entradas		Salidas		
EN	D	Q	~Q	
↑	0	0	1	Reset
↑	1	1	0	Set

5.2.3 Flip-Flop J-K

El Flip-Flop J-K, es un circuito secuencial cuyo comportamiento es idéntico al latch S-R en las operaciones de ***Set***, ***Reset*** y No cambio. La ventaja del Flip-Flop J-K es que es disparado por flanco y además no tiene estado no válido. La implementación es la siguiente.

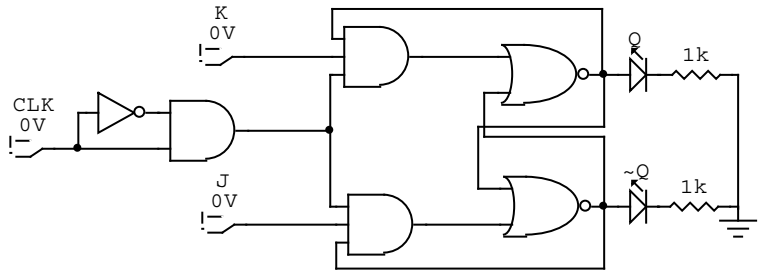


Figura 5.11 Flip-Flop J-K

Este Flip-Flop tiene como característica el hecho que en el estado J=1 y K=1, las salidas Q y ~Q basculan, es decir, si esta en 0 pasa a 1 y si esta en 1 pasa a 0.

La tabla de verdad del Flip-Flop J-K es:

Tabla 5.6 Tabla de verdad de Flip-Flop J-K

Entradas			Salidas		
CLK	J	K	Q	~Q	
↑	0	0	Q	~Q	No cambio
↑	0	1	1	0	Set
↑	1	0	0	1	Reset
↑	1	1	~Q	Q	Basculación

El símbolo de un Flip-Flop J-K, puede ser con flanco de subida o flanco de bajada. La siguiente figura muestra estos símbolos respectivamente.

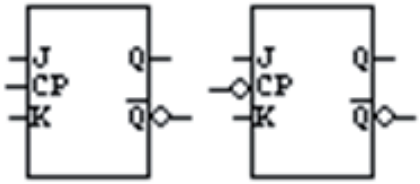


Figura 5.12 Símbolo del Flip-Flop J-K

5.2.4 Flip-Flop J-K con entradas asíncronas

Las entradas asíncronas son aquellas que no dependen de un pulso de reloj para afectar los estados de las salidas. En un Flip-Flop, puede haber dos entradas asíncronas llamadas *Clear* y *Preset*. Generalmente estas entradas son activas a nivel bajo. Entonces, cuando se coloca un 0 en la entrada *Clear*, ésta se activa y produce un 0 en la salida Q sin importar los valores de J y K. Igualmen-

te cuando se coloca un 0 en la entrada *Preset*, ésta se activa y produce un 1 en la salida Q sin importar los valores de J y K. La implementación es la siguiente.

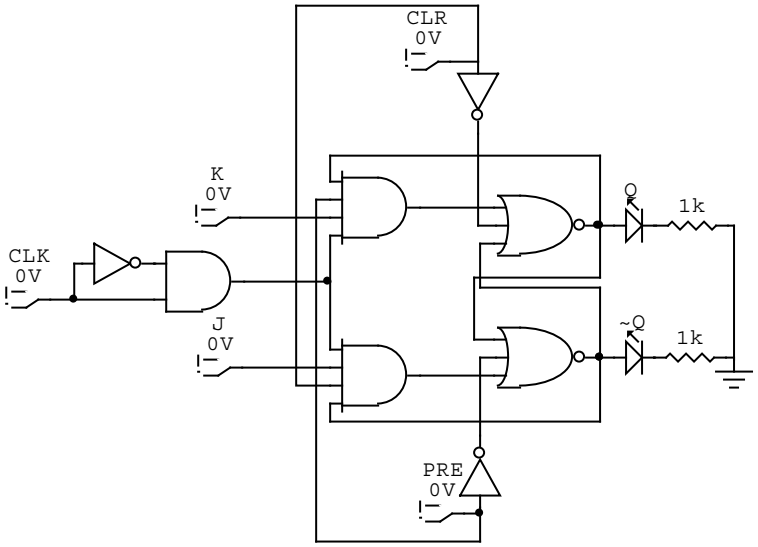


Figura 5.13 Flip-Flop J-K con entradas asíncronas

El símbolo de un Flip-Flop J-K con entradas asíncronas está representado en la siguiente figura:

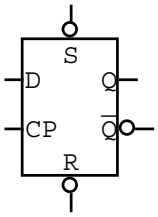


Figura 5.14 Símbolo del Flip-Flop J-K con entradas asíncronas

5.3 TEMPORIZADOR 555

El temporizador 555 es un circuito bastante utilizado que puede ser configurado de dos modos distintos, como monoestable o como aestable. Un multivibrador aestable, no tiene estado estable, por consiguiente varía constantemente, es decir, este circuito oscila entre dos estados correspondientes al 0 lógico (0 voltios DC) y el 1 lógico (5 voltios DC).

La figura 5.15 muestra un diagrama con los elementos del temporizador. Los comparadores son dispositivos cuyas salidas están a nivel alto cuando el voltaje en la entrada positiva es mayor que el voltaje en la entrada negativa y están a nivel bajo cuando el voltaje en la entrada negativa es mayor que el voltaje en

la entrada positiva. Hay un divisor de voltaje formado por tres resistencias de 5 KΩ que generan un nivel de disparo de 1/3Vcc y un nivel de umbral de 2/3 de Vcc. Hay un Latch que permite almacenar el estado mientras el temporizador posee niveles bajos en las salidas de los comparadores.

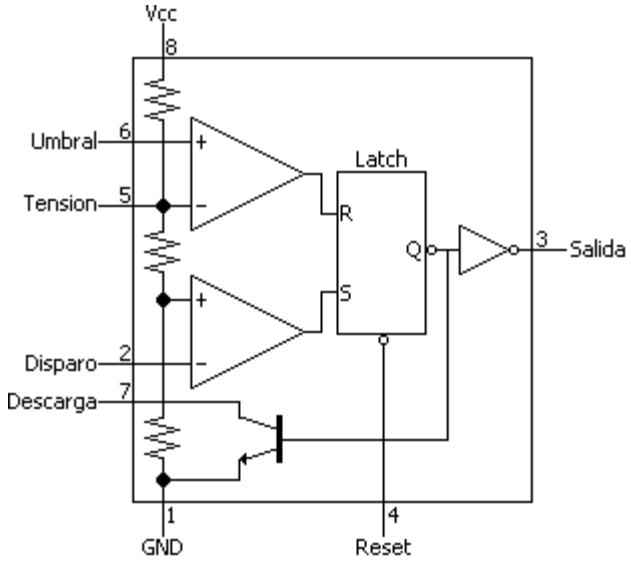


Figura 5.15 Diagrama de Temporizador 555

5.3.1 Temporizador 555 configurado como aestable

La figura 5.16 muestra la configuración de este circuito como aestable. En esta configuración debe conectarse dos resistencias (R1 y R2) y un condensador (C1). Las entradas disparo y umbral deben conectarse a condensador C1. Los elementos R1, R2 y C1, conforman la red de temporización que determina la frecuencia de oscilación.

Para comprender el funcionamiento, el temporizador cumple los siguientes eventos.

Al conectar el circuito, el condensador C1 está descargado, por consiguiente el voltaje en el pin 2 y 6 está en 0V, entonces la salida del comparado B, está a nivel alto y la salida del comparador A está en alto. El latch hace un set. La salida ~Q está en nivel bajo colocando el transistor en estado de corte. La salida del temporizador está en nivel alto.

El condensador C1 empieza a cargarse a través de las resistencias R1 y R2. Cuando el voltaje en el condensador supera 1/3Vcc, la salida del comparador B pasa a nivel bajo. En ese momento en el latch hay un estado de no cambio.

El condensador C1 continua cargándose, cuando el voltaje en el condensador supera 2/3Vcc, el comparado A cambia a nivel alto. El latch hace un ***reset***. La salida pasa a nivel bajo. El transistor queda en estado de saturación, entonces el condensador empieza a descargarse a través de la resistencia R2 y del transistor.

En el momento que el condensador se descarga y su voltaje es inferior a 2/3Vcc, el comparador A pasa a nivel bajo. El latch está en estado de no cambio.

Cuando el condensador queda con un voltaje inferior a 1/3Vcc, el comparador B queda en nivel alto y el latch hace un ***set***. La salida del temporizador pasa a nivel alto. El transistor queda en estado de corte. El condensador empieza a cargarse nuevamente a través de las resistencias R1 y R2.

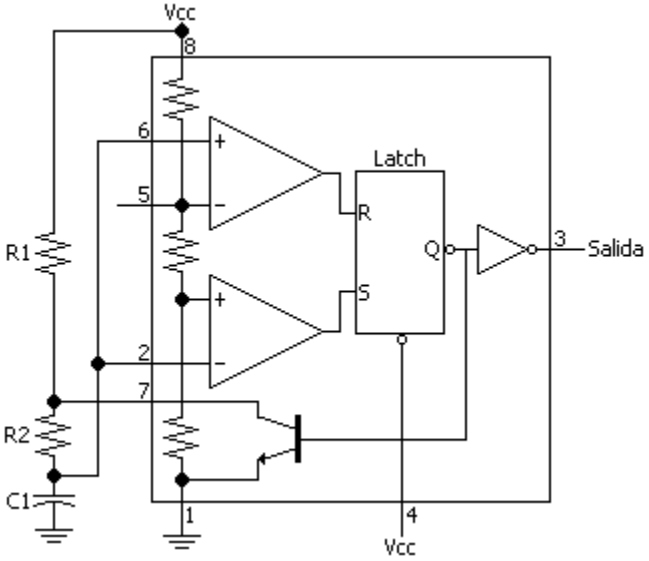


Figura 5.16 Diagrama de Temporizador 555 configurado como aestable

La frecuencia de oscilación del temporizador está dada por la siguiente fórmula:

$$f = \frac{1.4}{(R_1 + 2R_2)C_1}$$

El ciclo de trabajo en este circuito, se obtiene como una relación entre las resistencias R1 y R2, dado que el condensador se carga a través de R1 y R2 y se descarga a través de R2. Entonces el ciclo de trabajo se obtiene mediante la siguiente fórmula:

$$ciclo\ de\ trabajo = \left(\frac{R_1 + R_2}{R_1 + 2R_2} \right) 100\%$$

Se puede apreciar que el ciclo de trabajo nunca será de 50%, esto indica que el tiempo en nivel alto es diferente que el tiempo en nivel bajo.

Para obtener un ciclo de trabajo del 50%, es necesario lograr que el condensador se cargue por una resistencia y descargue por la otra. Se puede obtener colocando un diodo en paralelo a R2 con el fin de que el condensador se cargue por la resistencia R1 y a través del diodo cuya resistencia ideal es 0 y se descargue a través de R2. Entonces con esta configuración, la frecuencia y el ciclo de trabajo son:

$$f \approx \frac{1.4}{(R_1 + R_2)C_1}$$

$$\text{ciclo de trabajo} = \left(\frac{R_1}{R_1 + R_2}\right)100\%$$

5.4 DIVISOR DE FRECUENCIA

Una de las aplicaciones de los Flip-Flops es el divisor de frecuencia. El objetivo es reducir a la mitad la frecuencia de una señal periódica. Si se considera un Flip-Flop J-K y se conectan sus entradas J=K=1, la salida Q corresponderá a una señal periódica con la mitad de la frecuencia de la señal original. La implementación es la siguiente:

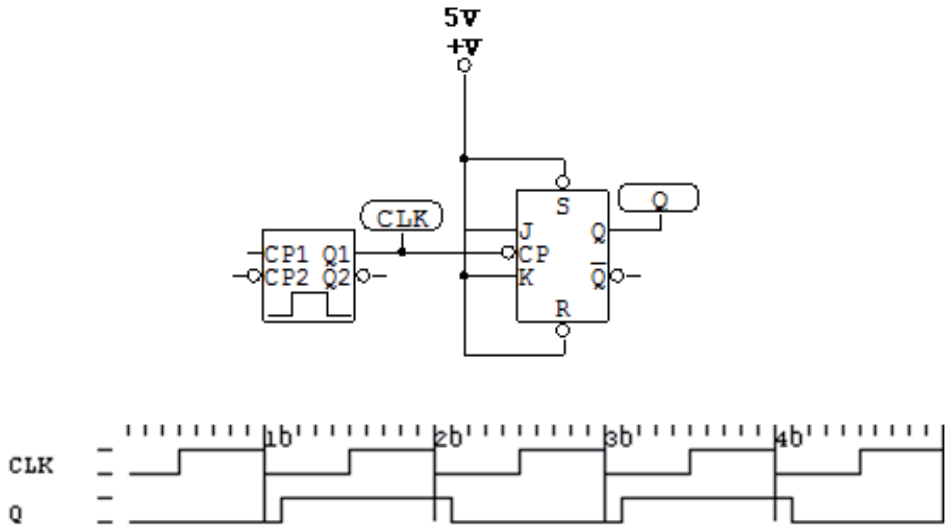


Figura 5.17 Divisor de frecuencia

En el oscilograma anterior, se puede apreciar que para cada flanco de bajada del reloj, hay una basculación que se observa en Q, un instante después de

dado el tiempo de propagación del Flip-Flop. Para este ejemplo, la frecuencia del reloj equivale a:

$$f_{clk} = \frac{1}{10\text{ms}} = 100\text{Hz}$$

$$f_{Q} = \frac{1}{20\text{ms}} = 50\text{Hz}$$

Ejercicios Propuestos

1. Calcular los elementos para obtener una frecuencia de 1 Hz en un temporizador 555 e implementar el resultado.

2. Calcular los elementos para obtener una frecuencia de 100 Hz en un temporizador 555 e implementar el resultado.

3. Calcular los elementos para obtener una frecuencia de 1 Hz en un temporizador 555 con ciclo de trabajo de 50 % e implementar el resultado.

4. Calcular los elementos para obtener una frecuencia de 100 Hz en un temporizador 555 con ciclo de trabajo de 50 % e implementar el resultado.

5. Con base en un temporizador configurado a 100 Hz, implementar un divisor de frecuencia para obtener una señal de 25 Hz

Capítulo 6
Contadores

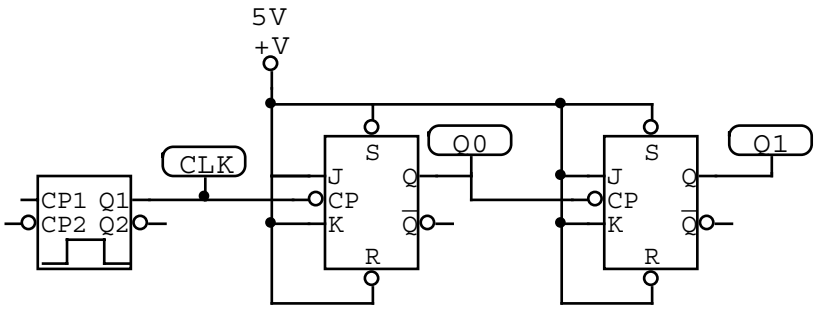
Un contador es un circuito secuencial el cual cambia de estado de acuerdo con una secuencia establecida por el diseño. Un contador está construido con base en Flip-Flops. El número de Flip-Flops utilizados indica el número de bits del contador, es decir, cada Flip-Flop representa un bit dentro de la secuencia de conteo.

6.1 CONTADOR ASÍNCRONO

El término asíncrono indica que los eventos no poseen una relación temporal fija entre ellos y que no necesariamente ocurren en el mismo instante de tiempo. Esto indica que en un contador asíncrono los Flip-Flops no comparten la misma señal de reloj.

6.1.1 Contador asíncrono binario de 2 bits

Para obtener un contador asíncrono, se debe usar 2 Flip-Flops J-K flanco de bajada. Se debe conectar de la misma forma que un divisor de frecuencia. Entonces se debe conectar la señal de reloj a la entrada de reloj del primer Flip-Flop cuya salida se considera Q_0. Esta salida entonces será el reloj del siguiente Flip-Flop cuya salida se denominará Q_1. Las entradas J y K deben estar en 1 lógico. En la figura 6.1 se muestra la implementación del contador y los oscilogramas que dan como resultado de su funcionamiento.



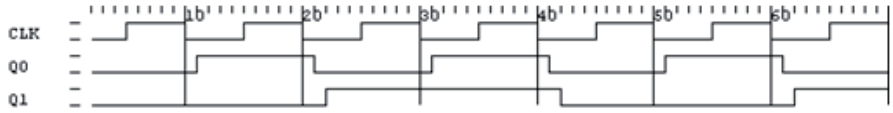


Figura 6.1 Contador asíncrono de 2 bits

En los oscilogramas, se puede apreciar que en cada flanco de bajada del reloj, báscula Q_0 y en cada flanco de bajada de Q_0, báscula Q_1. Analizando los valores de Q_0 y Q_1 en cada período de reloj, se nota que las salidas Q_0 y Q_1 forman estados que se pueden representar en una tabla denominada *tabla de secuencia* (véase tabla 6.1). En la figura 6.1 se muestra que en cada basculación existe un tiempo de retardo de propagación que equivale a 30 nseg que es el tiempo de retardo de un Flip-Flop J-K.

Tabla 6.1 Tabla de secuencia de contador asíncrono de 2 bits

CLK	Q_1	Q_0
↓	0	0
↓	0	1
↓	1	0
↓	1	1

6.1.2 Contador Asíncrono Binario de 4 bits

Para obtener un contador asíncrono de 4 bits, se debe usar 4 Flip-Flops J-K flanco de bajada. La implementación es igual que la anterior. La figura 6.2 muestra la implementación del contador y los oscilogramas que dan como resultado de su funcionamiento.

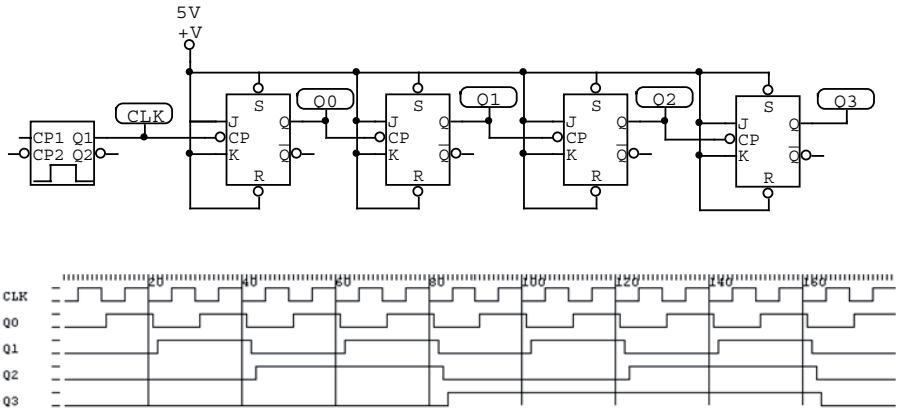


Figura 6.2 Contador asíncrono de 4 bits

En los oscilogramas, se puede apreciar que en cada flanco de bajada del reloj, báscula Q_0 y en cada flanco de bajada de Q_0, báscula Q_1 y sucesivamente. Analizando los valores de Q_0, Q_1, Q_2 y Q_3 en cada período de reloj, se nota que las salidas Q_0, Q_1, Q_2 y Q_3 forman estados que se pueden representar en una tabla de secuencia.

Tabla 6.2 Tabla de secuencia de contador asíncrono de 4 bits

CLK	Q_3	Q_2	Q_1	Q_0
↓	0	0	0	0
↓	0	0	0	1
↓	0	0	1	0
↓	0	0	1	1
↓	0	1	0	0
↓	0	1	0	1
↓	0	1	1	0
↓	0	1	1	1
↓	1	0	0	0
↓	1	0	0	1
↓	1	0	1	0
↓	1	0	1	1
↓	1	1	0	0
↓	1	1	0	1
↓	1	1	1	0
↓	1	1	1	1

6.1.3 Contador Asíncrono BCD

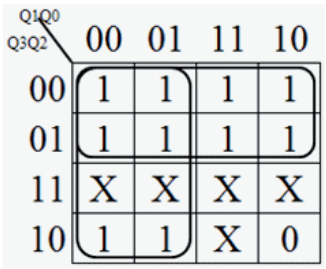
Para obtener un contador asíncrono BCD, se debe usar 4 Flip-Flops J-K flanco de bajada. La implementación es igual que la anterior. Sin embargo requiere un elemento adicional. La cuenta debe hacerse hasta el estado 1001_2 lo cual indica que no se desea la presencia del estado 1010_2. La solución es enviar una activación del ***Clear*** cuando se presente este estado, de esta forma el estado siguiente del 1001_2 sería el 0000_2.

Para obtener estos resultados es necesario encontrar una expresión Booleana que permita obtener los resultados deseados. Para el diseño se debe partir de la tabla de secuencia del contador considerando el valor del CLR.

Tabla 6.3 Tabla de secuencia de contador asíncrono BCD de 4 bits

Q_3	Q_2	Q_1	Q_0	CLR
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0

Para obtener la expresión del CLR se puede hacer un mapa de Karnaugh.



$CLR = \overline{B_3} \cdot \overline{B_1}$

$CLR = \overline{B_3 B_1}$

La figura 6.3 muestra la implementación del contador y los oscilogramas que dan como resultado de su funcionamiento.

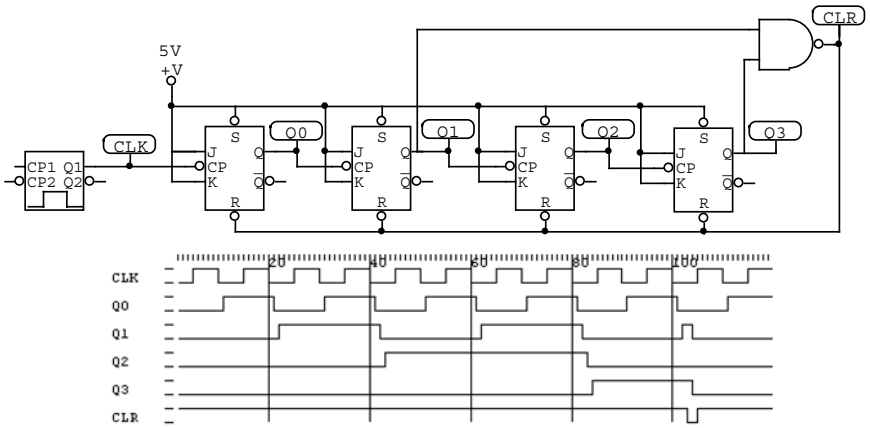


Figura 6.3 Contador asíncrono BCD

En los oscilogramas, se puede apreciar que en cada flanco de bajada del reloj, báscula Q_0 y en cada flanco de bajada de Q_0, báscula Q_1 y sucesivamente. Sin embargo, en el momento es que la combinación de las salidas dan el estado 1010_2 hay un *clear* asíncrono a todos los Flip-Flops regresando rápidamente la señal de *clear* a nivel alto. Este pequeño impulso que se presenta en la línea de *clear* se denomina ***glitch***, ya que se considera un impulso no deseado. Sin embargo, esta es la única forma de implementar contadores de modulo diferente de 2^n con un contador asíncrono. En la salida Q_1 también se presenta un ***glitch***, en el instante en que el contador tiene el valor 1010_2.

6.1.4 Contador asíncrono binario de 4 bits descendente

Para obtener un contador asíncrono de 4 bits descendente, se debe usar 4 Flip-Flops J-K flanco de subida. En este caso, la salida Q de un Flip-Flop, se convierte en el reloj del siguiente Flip-Flop. La implementación es exactamente igual al contador asíncrono binario de 4 bits, cambiando el flanco del reloj.

Otra forma de hacer la implementación, es usar Flip-Flops flanco de bajada, pero conectando la salida Q negado de un Flip-Flop al reloj del siguiente Flip-Flop. Esta segunda opción, suele ser más conveniente debido a que en el mercado se encuentra con mayor frecuencia Flip-Flops flanco de bajada.

La figura 6.4 muestra la implementación del contador y los oscilogramas que dan como resultado de su funcionamiento.

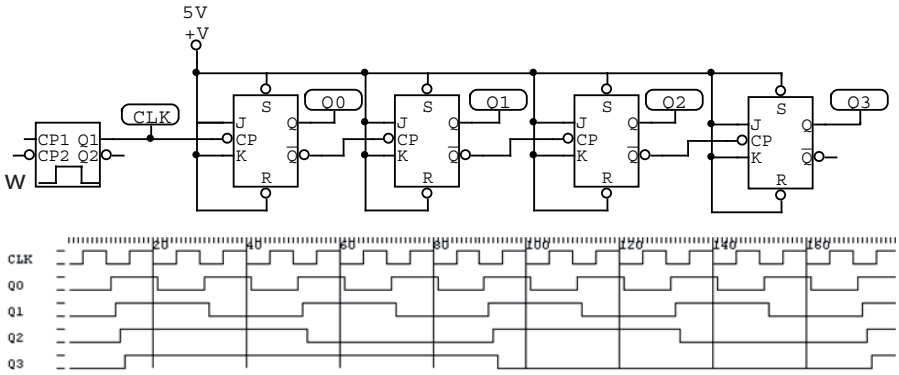


Figura 6.4 Contador asíncrono de 4 bits descendente

En los oscilogramas, se puede apreciar que en cada flanco de bajada del reloj, bascula Q_0 y en cada flanco de subida de Q_0, bascula Q_1 y sucesivamente. Analizando los valores de Q_0, Q_1, Q_2 y Q_3 en cada período de reloj, se nota que las salidas Q_0, Q_1, Q_2 y Q_3 forman estados que se pueden representar en una tabla de secuencia (véase tabla 6.4)

Tabla 6.4 Tabla de secuencia de contador asíncrono de 4 bits descendente

CLK	Q_3	Q_2	Q_1	Q_0
↓	0	0	0	0
↓	1	1	1	1
↓	1	1	1	0
↓	1	1	0	1
↓	1	1	0	0
↓	1	0	1	1
↓	1	0	1	0
↓	1	0	0	1
↓	1	0	0	0
↓	0	1	1	1
↓	0	1	1	0
↓	0	1	0	1
↓	0	1	0	0
↓	0	0	1	1
↓	0	0	1	0
↓	0	0	0	1

6.1.5 Contador Asíncrono Binario de 4 bits ascendente / descendente

Para obtener un contador asíncrono de 4 bits ascendente - descendente, se debe usar 4 Flip-Flops J-K flanco de bajada. Para este contador, es necesario tener una entrada adicional que permita seleccionar la cuenta ascendente o descendente. Por medio de esta entrada, se desea seleccionar Q o Q negado para que se aplique al reloj del siguiente Flip-Flop. Entonces, considerando que cuando esta entrada se encuentre en 0 lógico, el contador cuente ascendente y cuando se encuentre en 1 lógico el contador cuenta descendente, se plantea el siguiente diseño de la tabla 6.5.

Tabla 6.5 Tabla de control para contador asíncrono ascendente descendente

AD	Q	CLK
0	0	0
0	1	1
1	0	1
1	1	0

La tabla anterior plantea que cuando la entrada AD sea 0, el contador cuenta ascendente y para ello se debe aplicar al reloj del siguiente Flip-Flop la línea Q. Pero cuando la entrada AD sea 1, el contador cuenta descendente y para ello se debe aplicar al reloj del siguiente Flip-Flop la línea Q negado.

Entonces se obtiene la siguiente expresión Booleana.

$$CLK = \overline{AD}.Q + AD.\overline{Q}$$
$$CLK = AD \oplus Q$$

Con base en el resultado anterior, se puede concluir que al reloj del siguiente Flip-Flop, se debe aplicar una XOR entre la entrada AD y la salida Q. Esto se debe aplicar para los Flip-Flops 1, 2 y 3, debido a que el reloj del Flip-Flop 0 se encuentra conectado a la salida del temporizador.

La figura 6.5, en la página 126, muestra la implementación del contador y los oscilogramas que dan como resultado de su funcionamiento.

En los oscilogramas, se puede apreciar que mientras la entrada AD se encuentra en 1 lógico, el contador cuenta de forma descendente. En el momento en que la entrada es 0 lógico, el contador pasa a contar ascendente. Analizando los valores de Q_0, Q_1, Q_2 y Q_3 en cada período de reloj y con base en la entrada AD, se nota que las salidas Q_0, Q_1, Q_2 y Q_3 forman estados que se pueden representar en una tabla de secuencia.

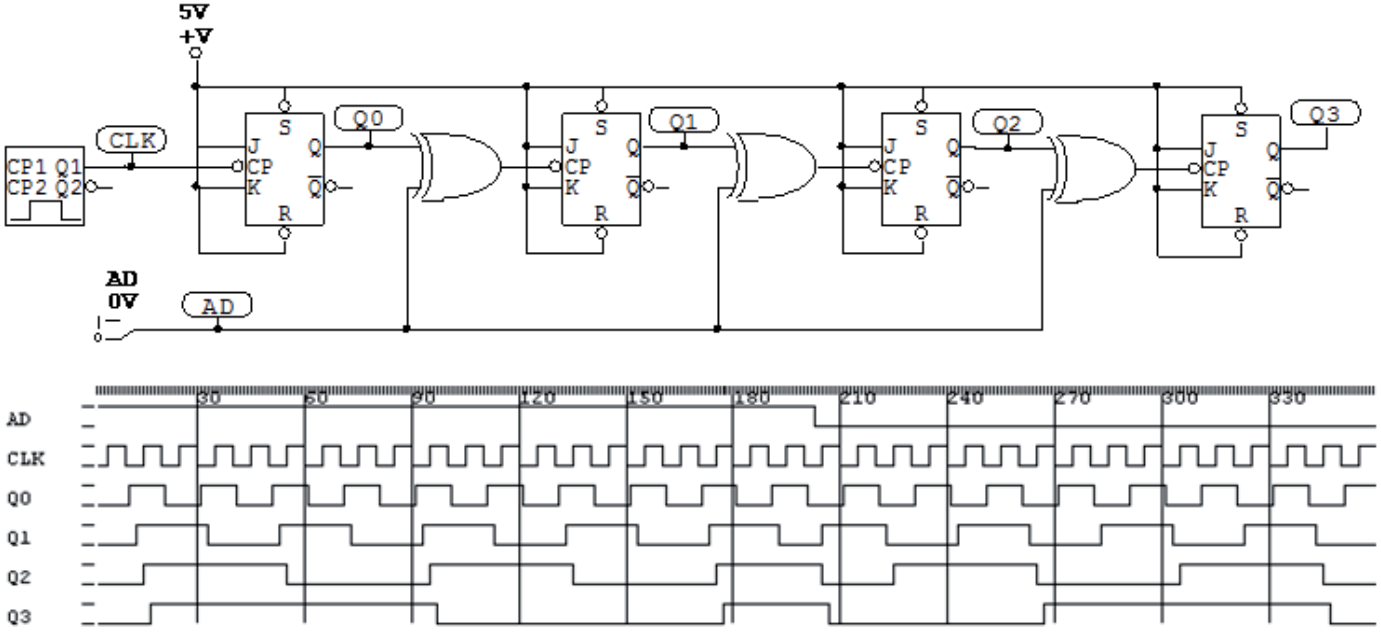


Figura 6.5 Contador asíncrono de 4 bits ascendente descendente

Tabla 6.6 Tabla de secuencia de contador asíncrono de 4 bits ascendente-descendente

CLK	AD	Q_3	Q_2	Q_1	Q_0
↓	1	0	0	0	0
↓	1	1	1	1	1
↓	1	1	1	1	0
↓	1	1	1	0	1
↓	1	1	1	0	0
↓	1	1	0	1	1
↓	1	1	0	1	0
↓	1	1	0	0	1
↓	1	1	0	0	0
↓	1	0	1	1	1
↓	1	0	1	1	0
↓	1	0	1	0	1
↓	1	0	1	0	0
↓	1	0	0	1	1
↓	1	0	0	1	0
↓	1	0	0	0	1
↓	0	0	0	0	0
↓	0	0	0	0	1
↓	0	0	0	1	0
↓	0	0	0	1	1
↓	0	0	1	0	0
↓	0	0	1	0	1
↓	0	0	1	1	0
↓	0	0	1	1	1
↓	0	1	0	0	0
↓	0	1	0	0	1
↓	0	1	0	1	0
↓	0	1	0	1	1
↓	0	1	1	0	0
↓	0	1	1	0	1
↓	0	1	1	1	0
↓	0	1	1	1	1

6.2 CONTADOR SÍNCRONO

El término síncrono indica que los eventos poseen una relación temporal fija entre ellos, es decir, que ocurren en el mismo instante de tiempo. Esto indica que en un contador síncrono los Flip-Flops siempre comparten la misma señal de reloj.

6.2.1 Contador Síncrono Binario de 2 bits

Para obtener un contador síncrono, se debe usar 2 Flip-Flops J-K. Como todo el reloj es común, no importa si es flanco de subida o bajada en los Flip-Flops, pero todos los Flip-Flops deben ser iguales. Entonces se debe conectar la señal de reloj a todos los Flip-Flops. Las entradas J y K del Flip-Flop cuya salida en Q_0, es decir, J_0 y K_0 deben ir conectados a Vcc, esto va a permitir que esta salida siempre bascule. Luego se conecta Q_0 a las entradas J_1 y K_1. La figura 6.6 muestra la implementación del contador y los oscilogramas que dan como resultado de su funcionamiento.

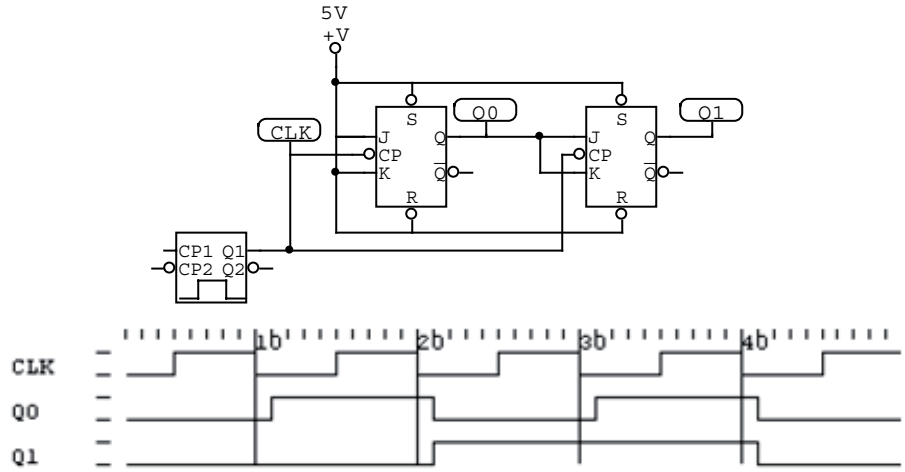


Figura 6.6 Contador síncrono de 2 bits

En los oscilogramas, se puede apreciar el mismo comportamiento que el contador asíncrono de 2 bits, sin embargo, esta implementación tiene una mejora radical. Todos los Flip-Flops actúan en el mismo instante de tiempo, esto indica que el retardo de propagación de un estado a otro siempre es el mismo sin importar el estado en que se encuentre.

6.2.2 Contador Síncrono Binario de 4 bits

Para obtener un contador síncrono de 4 bits, se debe usar 4 Flip-Flops J-K. La implementación es igual que la anterior, es decir que el Flip-Flop cuya salida es Q_2 tiene en sus entradas J_2 y K_2 una AND entre Q_0 y Q_1. La figura 6.7 muestra la implementación del contador y los oscilogramas que dan como resultado de su funcionamiento.

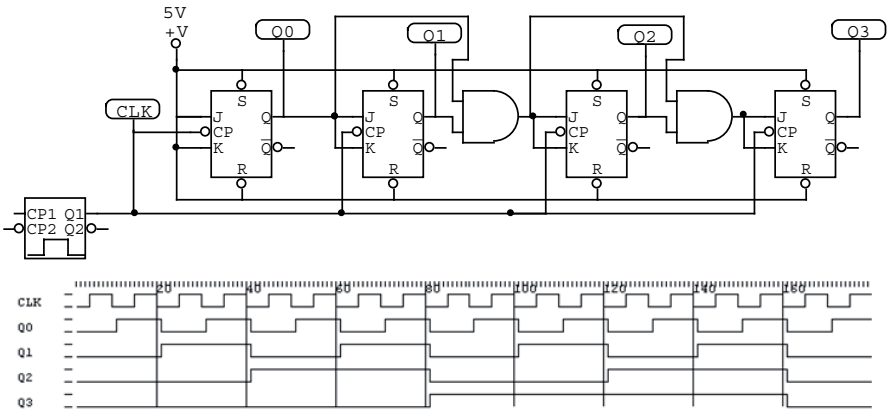


Figura 6.7 Contador síncrono de 4 bits

En los oscilogramas, se puede apreciar mismo comportamiento que el contador asíncrono de 4 bits, sin embargo, esta implementación tiene una mejora radical. Todos los Flip-Flops actúan en el mismo instante de tiempo, esto indica que el retardo de propagación de un estado a otro siempre es el mismo sin importar el estado en que se encuentre.

6.3 CONTADOR SÍNCRONO ASCENDENTE DESCENDENTE

Un contador síncrono ascendente descendente, es aquel capaz de contar en una secuencia determinada en cualquier dirección. Suponiendo un contador binario de 3 bits, si es sólo ascendente éste contaría desde 0 a 7. Si se tiene un contador ascendente descendente, este contador puede contar de 0 a 7 y de 7 a 0, controlando la dirección por una entrada adicional.

En un contador síncrono, las entradas de J y K de cada Flip-Flop, dependen de las salidas Q de los Flip-Flops anteriores produciendo un conteo ascendente. Para generar un conteo descendente, las entradas J y K de cada Flip-Flop, deben depender de las salidas Q negado de los Flip-Flops.

Para diseñar un contador ascendente descendente binario de 3 bits, se debe tener en cuenta la siguiente tabla de secuencia.

Tabla 6.7 Tabla de secuencia de contador asíncrono de 4 bits ascendente descendente

CLK	Asc/Des	Q_2	Q_1	Q_0
↓	0	0	0	0
↓	0	0	0	1
↓	0	0	1	0
↓	0	0	1	1
↓	0	1	0	0
↓	0	1	0	1
↓	0	1	1	0
↓	0	1	1	1
↓	0	0	0	0
↓	1	0	0	0
↓	1	1	1	1
↓	1	1	1	0
↓	1	1	0	1
↓	1	1	0	0
↓	1	0	1	1
↓	1	0	1	0
↓	1	0	0	1
↓	1	0	0	0
↓	1	1	1	1

En esta tabla se puede apreciar que si la cuenta es ascendente, el estado siguiente de 111_2 es 000_2. Si la cuenta es descendente el estado siguiente de 000_2 es 111_2.

Con base en los conceptos anteriores, la implementación del contador se presenta en la figura 6.8 (*ver página siguiente)*

En el oscilograma se observa que mientras la entrada A/D se encuentra en 0, el contador cuenta de forma ascendente y mientras la entrada A/D se encuentra en 1, el contador cuenta de forma descendente.

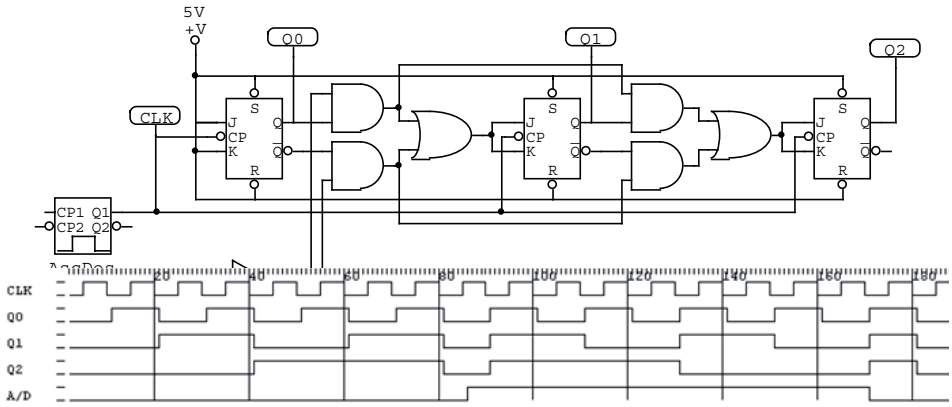


Figura 6.8 Contador Ascendente Descendente de 3 bits

6.4 DISEÑO DE CONTADORES SÍNCRONOS

Basado en el concepto de contador síncrono, es posible elaborar diseños para obtener una secuencia determinada. Esta secuencia, puede llegar a ser incluso una secuencia aleatoria.

Para realizar el diseño del contador, es necesario seguir una serie de pasos. Para explicar el proceso de diseño, se plantea un contador de código Gray.

Paso 1 de diseño de contador: Tabla de Secuencia

Es necesario determinar la tabla de secuencia. La tabla de secuencia debe especificar en estricto orden los estados que tendrá el contador. Hay que tener en cuenta que el siguiente estado del último será siempre el primero. Es necesario también tener en cuenta que un estado tendrá únicamente un siguiente estado. La tabla de secuencia también determina el número de bits del contador y a su vez determina el número de Flip-Flops necesarios para el contador. Por cada bit en la tabla de secuencia, se requiere un Flip-Flop en la implementación del contador. Para el caso del contador de código Gray se usará 4 bits que implica usar 4 Flip-Flops. La tabla de secuencia del contador de código Gray es la siguiente:

Tabla 6.8 Tabla de secuencia de contador código Gray

CLK	Q_3	Q_2	Q_1	Q_0
↓	0	0	0	0
↓	0	0	0	1
↓	0	0	1	1
↓	0	0	1	0
↓	0	1	1	0
↓	0	1	1	1
↓	0	1	0	1
↓	0	1	0	0
↓	1	1	0	0
↓	1	1	0	1
↓	1	1	1	1
↓	1	1	1	0
↓	1	0	1	0
↓	1	0	1	1
↓	1	0	0	1
↓	1	0	0	0

Paso 2 de diseño de contador: Tablas de transición de estados del Flip-Flop

Para la implementación del contador, es necesario escoger un tipo de Flip-Flop. Generalmente se elige Flip-Flop J-K gracias a sus 4 posibles estados. La tabla de transición de estados del Flip-Flop, determina qué valores se requieren en las entradas J y K para obtener una transición de estado actual a un estado futuro. Un estado actual hace referencia al valor de la salida Q del Flip-Flop un instante antes de efectuar un flanco de reloj. Un estado futuro hace referencia al valor de la salida Q del Flip-Flop un instante después de efectuar un flanco de reloj. Entonces la tabla de transición de estados del Flip-Flop J-K es la siguiente:

Tabla 6.9 Tabla de transición de estados del Flip-Flop J-K

Transiciones de salida		Entradas	
Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

La tabla anterior expone los siguientes casos:

- Si el Flip-Flop tiene un estado presente Q_t=0 y se desea obtener un estado futuro Q_{t+1}=0 entonces se le debe proveer al Flip-Flop un estado de no cambio o un estado de reset. Esto indica que se le debe proveer un J=0, K=0 o J=0, K=1. Esto indica que necesariamente J debe ser 0 y K puede ser 0 o 1. Entonces se enuncia como J=0 y K=X.

- Si el Flip-Flop tiene un estado presente Q_t=0 y se desea obtener un estado futuro Q_{t+1}=1 entonces se le debe proveer al Flip-Flop un estado de set o un estado de basculación. Esto indica que se le debe proveer un J=1, K=0 o J=1, K=1. Esto indica que necesariamente J debe ser 1 y K puede ser 0 o 1. Entonces se enuncia como J=1 y K=X.

- Si el Flip-Flop tiene un estado presente Q_t=1 y se desea obtener un estado futuro Q_{t+1}=0 entonces se le debe proveer al Flip-Flop un estado de reset o un estado de basculación. Esto indica que se le debe proveer un J=0, K=1 o J=1, K=1. Esto indica que necesariamente K debe ser 1 y J puede ser 0 o 1. Entonces se enuncia como J=X y K=1.

- Si el Flip-Flop tiene un estado presente Q_t=1 y se desea obtener un estado futuro Q_{t+1}=1 entonces se le debe proveer al Flip-Flop un estado de no cambio o un estado de set. Esto indica que se le debe proveer un J=0, K=0 o J=1, K=0, lo que necesariamente se concluye es que K debe ser 0 y J puede ser 0 o 1. Entonces se enuncia como J=X y K=0.

Paso 3 de diseño de contador: Mapas de Karnaugh

Los mapas de Karnaugh se usan para determinar la lógica requerida para las entradas J y K de cada Flip-Flop del contador. Se debe utilizar un mapa de Karnaugh para la entrada J y un mapa de Karnaugh para la entrada K. Cada celda del mapa de Karnaugh representa un estado de la tabla de secuencia establecida para el diseño. En cada celda del mapa de Karnaugh, se coloca el valor que requiere el Flip-Flop para obtener el estado siguiente necesario. Por ejemplo: el primer estado de la tabla de secuencia es 0000_2 y el estado siguiente es 0001_2, entonces si se desea diseñar la salida Q_0, se debe hacer una mapa de Karnaugh para J_0 y un mapa de Karnaugh para K_0. Para el mapa de Karnaugh de J_0, se debe colocar en la celda que corresponde al valor 0000_2, lo que necesita J en un Flip-Flop J-K para que el estado futuro sea 1, dado un estado presente 0. Revisando la tabla de transición de estado del Flip-Flop J-K, se puede observar que el valor necesario es 1, entonces en la celda 0000 del Flip-Flop se coloca un 1. Se llena el mapa de Karnaugh en todas sus celdas. Se obtiene los siguientes mapas:

J_0

Q3Q2 \ Q1Q0	00	01	11	10
00	1	X	X	0
01	0	X	X	1
11	1	X	X	0
10	0	X	X	1

K_0

Q3Q2 \ Q1Q0	00	01	11	10
00	X	0	1	X
01	X	1	0	X
11	X	0	1	X
10	X	1	0	X

J_1

Q3Q2 \ Q1Q0	00	01	11	10
00	0	1	X	X
01	0	0	X	X
11	0	1	X	X
10	0	0	X	X

K_1

Q3Q2 \ Q1Q0	00	01	11	10
00	X	X	0	0
01	X	X	1	0
11	X	X	0	0
10	X	X	1	0

J_2

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	0	1
01	X	X	X	X
11	X	X	X	X
10	0	0	0	0

K_2

Q3Q2 \ Q1Q0	00	01	11	10
00	X	X	X	X
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

J_3

Q3Q2 \ Q1Q0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	X	X	X	X
10	X	X	X	X

K_3

Q3Q2 \ Q1Q0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	0	0
10	1	0	0	0

Paso 4 de diseño de contador: Expresiones Booleanas

Una vez se han construido los mapas de Karnaugh, se hace la simplificación para obtener las expresiones Booleanas para J y K de cada Flip-Flop.

$$J_0 = \overline{Q_3}\,\overline{Q_2}\,\overline{Q_1} + \overline{Q_3}Q_2Q_1 + Q_3Q_2\overline{Q_1} + Q_3\overline{Q_2}Q_1 \qquad K_0 = \overline{Q_3}\,\overline{Q_2}Q_1 + \overline{Q_3}Q_2\overline{Q_1} + Q_3Q_2Q_1 + Q_3\overline{Q_2}\,\overline{Q_1}$$

$$J_0 = \overline{Q_3}\left(\overline{Q_2}\,\overline{Q_1} + Q_2Q_1\right) + Q_3\left(Q_2\overline{Q_1} + \overline{Q_2}Q_1\right) \qquad K_0 = \overline{Q_3}\left(\overline{Q_2}Q_1 + Q_2\overline{Q_1}\right) + Q_3\left(Q_2Q_1 + \overline{Q_2}\,\overline{Q_1}\right)$$

$$J_0 = \overline{Q_3}\left(\overline{Q_2 \oplus Q_1}\right) + Q_3\left(Q_2 \oplus Q_1\right) \qquad K_0 = \overline{Q_3}\left(Q_2 \oplus Q_1\right) + Q_3\left(\overline{Q_2 \oplus Q_1}\right)$$

$$J_0 = \overline{Q_3 \oplus Q_2 \oplus Q_1} \qquad K_0 = Q_3 \oplus Q_2 \oplus Q_1$$

$$J_1 = \overline{Q_3}\,\overline{Q_2}Q_0 + Q_3Q_2Q_0 \qquad K_1 = \overline{Q_3}Q_2Q_0 + Q_3\overline{Q_2}Q_0$$

$$J_1 = Q_0\left(\overline{Q_3}\,\overline{Q_2} + Q_3Q_2\right) \qquad K_1 = Q_0\left(\overline{Q_3}Q_2 + Q_3\overline{Q_2}\right)$$

$$J_1 = Q_0\left(\overline{Q_3 \oplus Q_2}\right) \qquad K_1 = Q_0\left(Q_3 \oplus Q_2\right)$$

$$J_2 = \overline{Q_3}Q_1\overline{Q_0} \qquad K_2 = Q_3Q_1\overline{Q_0}$$

$$J_3 = Q_2\overline{Q_1}\,\overline{Q_0} \qquad K_2 = \overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}$$

Paso 5 de diseño de contador: Implementación

Se implementa el contador con base en las expresiones Booleanas obtenidas en el paso anterior. Éste se presenta en la siguiente figura (véase figura 6.9):

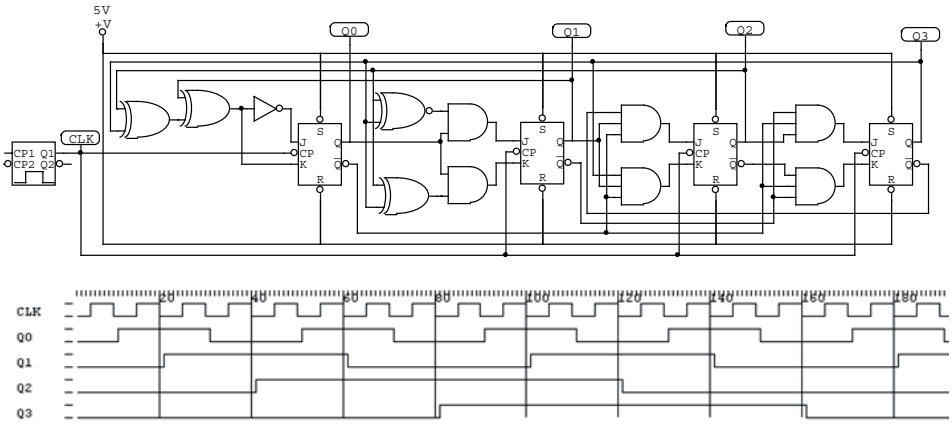


Figura 6.9 Contador síncrono código Gray de 4 bits

En la figura anterior se ha implementado los resultados de los mapas de Karnaugh para cada una de las entradas del Flip-Flop. Los oscilogramas muestran la secuencia descrita en la tabla de secuencia.

Paso 6 de diseño de contador: Pruebas

Se puede probar el funcionamiento del contador con los oscilogramas, sin embargo la lectura de éstos puede llegar a ser compleja. En la práctica la forma más adecuada de revisar el comportamiento del contador es usando un conversor binario–BCD, un decodificador manejador y dos displays que van a permitir visualizar los diferentes valores desde 0 hasta 15 decimal.

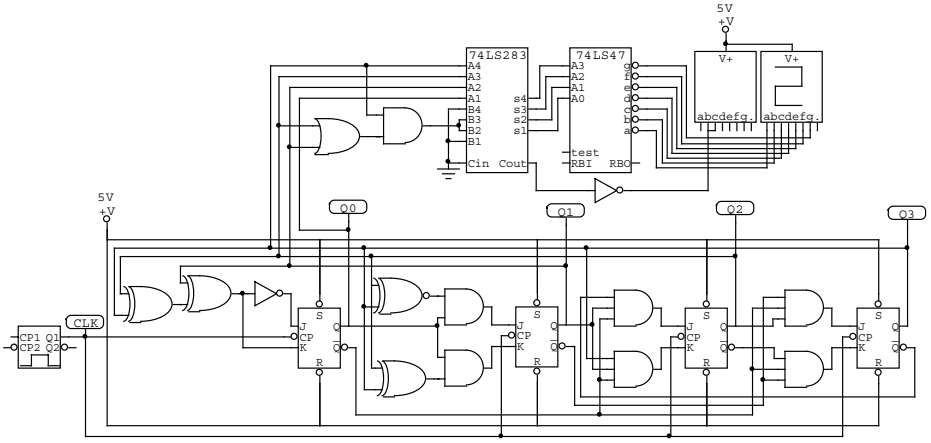


Figura 6.10 Contador síncrono código Gray de 4 bits con visualización

6.5 CONTADORES EN CASCADA

Los contadores en cascada son una implementación que permite conectar varios contadores para generar una secuencia de conteo de varios dígitos. Los contadores con los que se construye un contador en cascada, generalmente son contadores módulo 10, sin embargo, pueden ser de módulo menor. Adicionalmente, en un contador en cascada puede intervenir contadores de diferentes módulos. Un ejemplo puede ser un reloj digital. Para implementar un reloj digital, se requeriría un contador módulo 10 para las unidades de segundos, un contador módulo 6 para las decenas de segundos, un contador módulo 10 para las unidades de minutos, un contador módulo 6 para las decenas de minutos y un contador módulo 12 para las horas.

Para implementar un contador en cascada, es necesario partir de contadores previamente implementados preferiblemente síncronos. Para ello se puede hacer uso de diferentes circuitos integrados como el 74LS190 en figura 6.11:

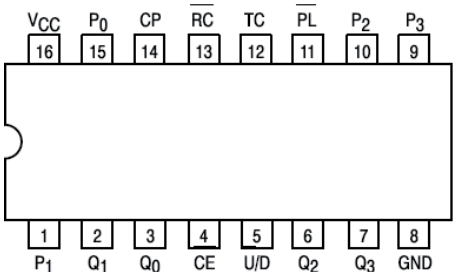


Figura 6.11 74LS190. Contador síncrono decimal asc/des con carga en paralelo

Por ejemplo, se puede implementar un contador módulo 100 ascendente/descendente con base en dos contadores módulo 10. Para ello, entonces se utilizaría el contador 74LS190. Este contador tiene una salida que es RC (Riple clock), la cual se coloca en 0, cuando el valor del contador es el mayor y su cuenta es ascendente o cuando el valor del contador es el menor y su cuenta es descendente.

Entonces se podría obtener la secuencia deseada conectando los relojes de los dos contadores a un oscilador y la salida RC del contador del dígito de menor peso o unidades a la entrada CE (*chip enable*) del contador del dígito de mayor peso o decenas.

La implementación se presenta en la figura 6.12 de la página siguiente.

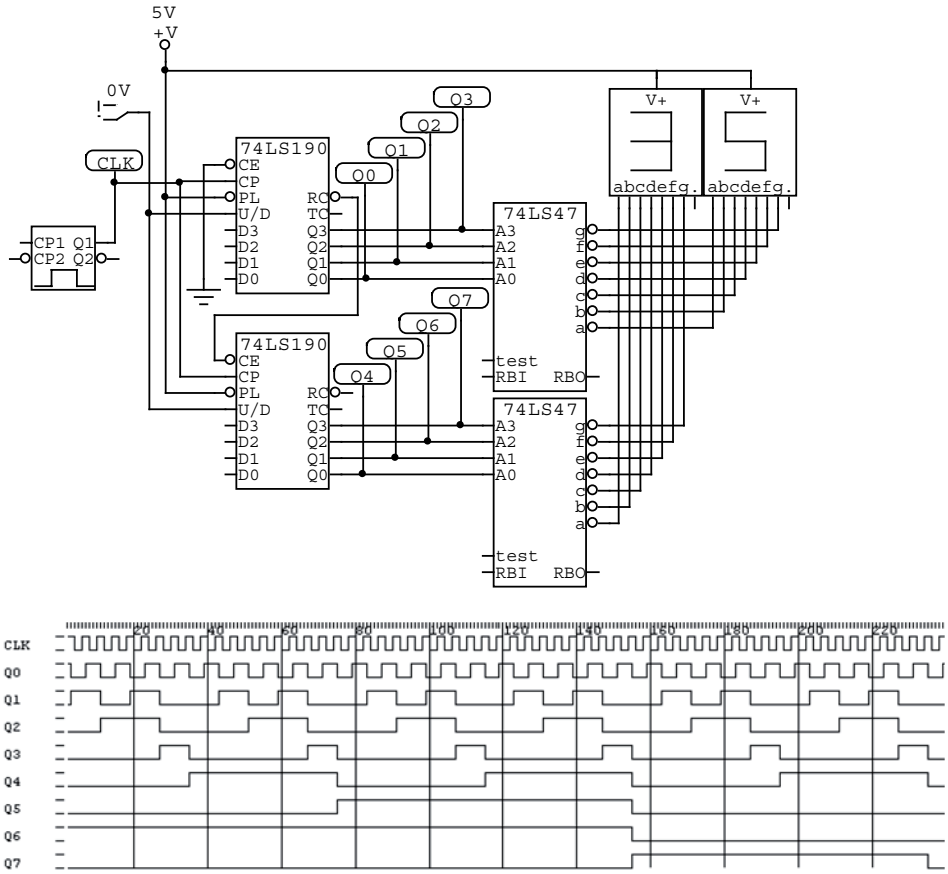


Figura 6.12 74LS190. Contador en cascada módulo 100

Ejercicios Propuestos

1. Diseñar e implementar un contador asíncrono módulo 12.

2. Diseñar e implementar un contador asíncrono descendente módulo 12.

3. Diseñar e implementar un contador asíncrono ascendente descendente módulo 10.

4. Diseñar e implementar un contador síncrono módulo 10.

5. Diseñar e implementar un contador síncrono con la siguiente secuencia: 0,2,4,6,8,10,12,14,15,13,11,9,7,5,3,1

6. Diseñar e implementar un contador síncrono con la siguiente secuencia: 0,1,2,3,15,14,13,12,4,5,6,7,11,10,9,8

7. Diseñar e implementar un contador síncrono con la siguiente secuencia: 0,1,2,3,4,5,6,7, 15,14,13,12,11,10,9,8

8. Diseñar e implementar un contador síncrono en cascada módulo 60.

9. Diseñar e implementar un contador síncrono en cascada módulo 100.

Capítulo 7

Registros

Los registros son circuitos secuenciales construidos a base de Flip-Flops los cuales permiten almacenamiento de información. Generalmente, los registros están construidos a base de Flip-Flops tipo D, debido a que siempre que a la entrada se coloca un 1 lógico, se desea obtener a la salida Q un 1 lógico, y siempre que a la entrada se coloca un 0 lógico, se desea obtener a la salida Q un 0 lógico. Un registro puede presentar diferentes funcionalidades definida por el desplazamiento de la información y la carga en paralelo de la información.

Los registros, pueden tener diferentes movimientos, los cuales dependen de su forma de entrada y salida de datos.

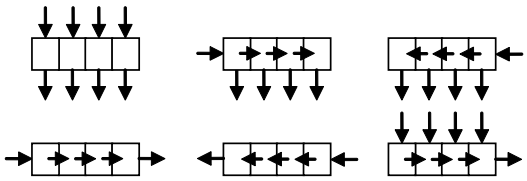


Figura 7.1 Movimientos de datos en un registro

En la figura anterior (véase figura 7.1) se muestran los diferentes movimientos de datos en un registro. La primera imagen corresponde a entrada en paralelo, salida en paralelo. La segunda imagen corresponde a entrada en serie con desplazamiento a la derecha, salida en paralelo. La tercera imagen corresponde a entrada en serie con desplazamiento a la izquierda, salida en paralelo. La cuarta imagen corresponde a entrada en serie con desplazamiento a la derecha, salida en serie. La quinta imagen corresponde a entrada en serie con desplazamiento a la izquierda, salida en serie. La sexta imagen corresponde a entrada en paralelo, salida en serie con desplazamiento a la derecha.

7.1 REGISTROS CON ENTRADA Y SALIDA EN PARALELO

Un registro con entrada y salida en paralelo, tiene como objetivo, capturar un dato en paralelo y colocarlo directamente en la salida del registro. La figura 7.2 presenta el diseño de dicho registro:

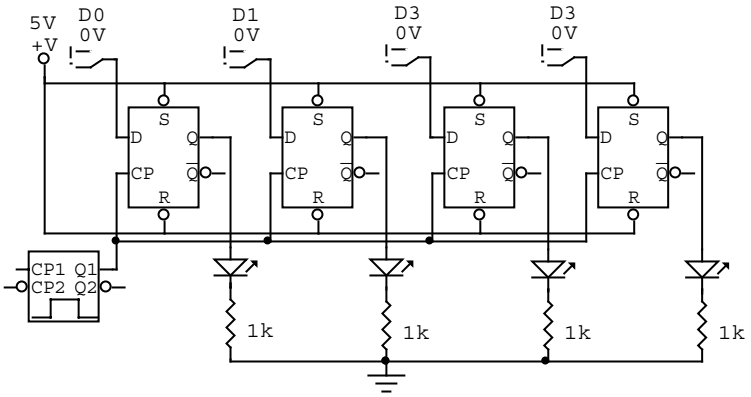


Figura 7.2 Registro con entrada y salida en paralelo

7.2 REGISTROS DE DESPLAZAMIENTO CON ENTRADA Y SALIDA EN SERIE

Un registro con entrada y salida en serie, tiene como objetivo, capturar un dato bit a bit dado una frecuencia. Con este tipo de implementaciones, la entrada y la salida del registro son sólo un bit.

Los registros de desplazamiento pueden tener entrada de datos en el bit de menor peso, de esta forma los bits a la salida se obtienen en el bit de mayor peso. Este tipo de desplazamiento, es un desplazamiento a la izquierda. Este desplazamiento se puede representar de la siguiente forma:

$Q3$	$Q2$	$Q1$	$Q0$
0	0	0	0
0	0	0	1
0	0	1	0
0	1	0	0

En la representación anterior, se inicializa las salidas Q en 0. Suponiendo que se ingresa por el bit de menor peso un uno, en el siguiente flanco de reloj, se desplaza hacia la izquierda todos los bits, obteniendo un desplazamiento por cada flanco de reloj.

La figura 7.3 presenta un registro de desplazamiento hacia la izquierda:

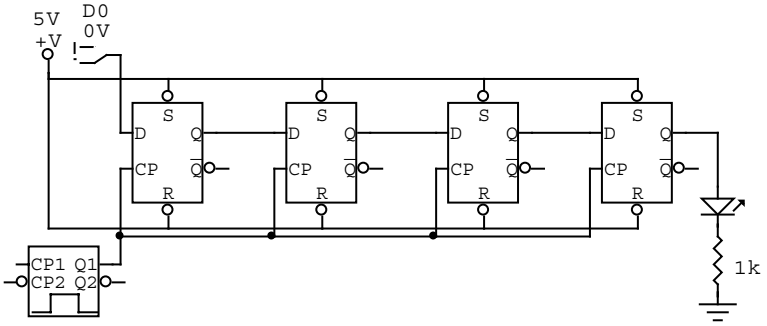


Figura 7.3 Registro de desplazamiento hacia la izquierda

Por otro lado, los registros de desplazamiento pueden tener entrada de datos en el bit de mayor peso, de esta forma los bits a la salida se obtienen en el bit de menor peso. Este tipo de desplazamiento, es un desplazamiento a la derecha. Este desplazamiento se puede representar de la siguiente forma:

$Q3$	$Q2$	$Q1$	$Q0$
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0

En la representación anterior, se inicializa las salidas Q en 0. Suponiendo que se ingresa por el bit de mayor peso un uno, en el siguiente flanco de reloj, se desplaza hacia la derecha todos los bits, obteniendo un desplazamiento por cada flanco de reloj.

La figura 7.4 presenta un registro de desplazamiento hacia la derecha:

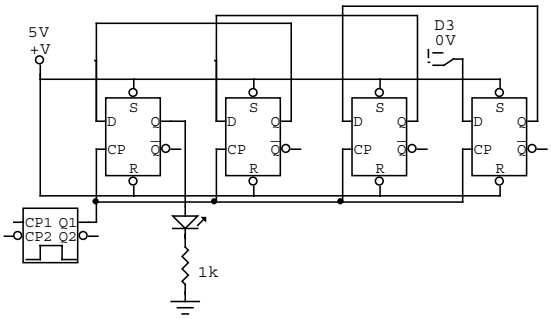


Figura 7.4 Registro de desplazamiento hacia la derecha

7.3 REGISTROS DE DESPLAZAMIENTO EN CASCADA

El registro 74194 contiene en su configuración diferentes posibilidades. Este registro puede hacer carga en paralelo, desplazamiento a la derecha y desplazamiento a la izquierda. Este circuito además es apropiado para hacer conexiones en cascada, de esa forma, puede implementarse registro de más de 4 bits.

La figura 7.5 muestra una implementación de un registro de 8 bits construido con base en registros 74194, los cuales son registros de 4 bits.

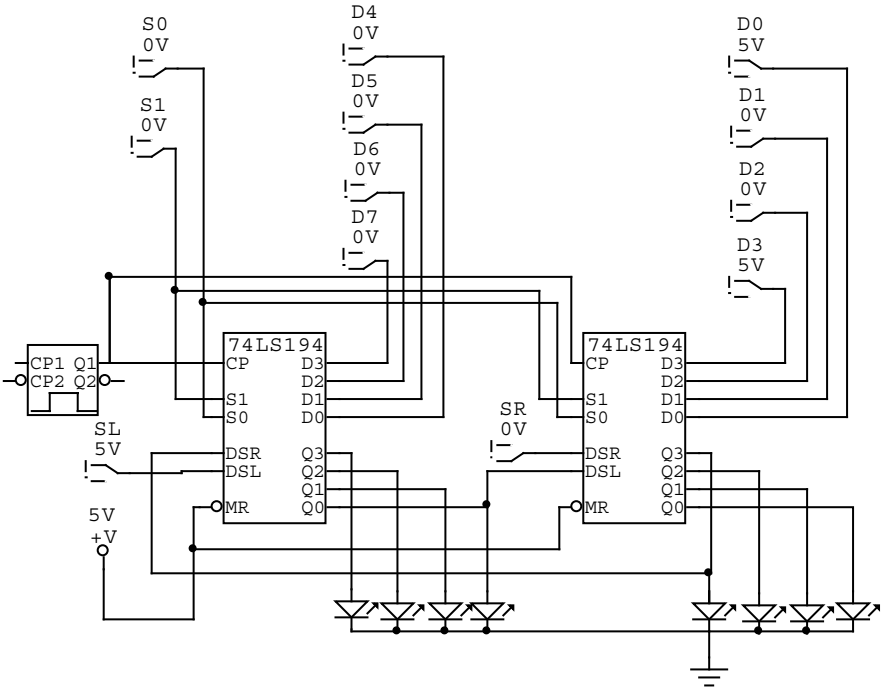


Figura 7.5 Registro de desplazamiento en cascada

Ejercicios Propuestos

1. Diseñar e implementar un registro de desplazamiento a la derecha y a la izquierda.

2. Diseñar e implementar un registro de desplazamiento a la derecha y carga en paralelo.

3. Diseñar e implementar un registro de desplazamiento a la derecha, a la izquierda y carga en paralelo.

4. Diseñar e implementar un registro de desplazamiento a la derecha y carga en paralelo de 8 bits.

Capítulo 8
Memorias

Una memoria es un dispositivo de almacenamiento de datos binarios. Las memorias almacenan bloques de datos generalmente de 8 bits que corresponden a un byte. Un byte está compuesto por dos bloques de 4 bits que reciben el nombre de *nibble*. Una unidad de información se denomina *palabra* y está compuesta por uno o más bytes.

Una memoria está formada por matrices de celdas. Una celda es un elemento de almacenamiento que puede almacenar un 1 o 0 lógico. La posición en una unidad de datos se denomina *dirección*. La dirección corresponde a la ubicación de una unida de datos o palabra dentro de la memoria.

La capacidad de una memoria, corresponde al número total de unidades de datos que puede almacenar. La operación de escritura en una memoria coloca los datos en una posición de memoria específica. La operación de lectura toma los datos de una posición de memoria específica.

8.1 LECTURA Y ESCRITURA

Los datos se colocan en la memoria a través del proceso de escritura y se toman de la memoria a través del proceso de lectura mediante un conjunto de líneas denominadas *bus de datos*. El bus de datos en una memoria es bidireccional, lo que indica que los datos pueden entrar a la memoria o salir de ella. En el caso de la memoria organizada en bytes, el bus de datos tiene 8 líneas que corresponden a 8 bits.

En las operaciones de lectura y escritura, se debe seleccionar una posición de memoria mediante una dirección que se debe colocar en un dato binario. Esta dirección se ubica en un conjunto de líneas denominadas *bus de direcciones*. El código de la dirección se decodifica mediante un circuito combinacional denominado *decodificador de instrucciones* y selecciona el registro en donde se debe almacenar o leer el dato. El número de líneas del bus de direcciones depende de la capacidad de la memoria. Esta capacidad está determinada por potencias base 2, debido a que dicha dirección se encuentra en binario. Por

ejemplo, si una memoria tiene un bus de datos de 8 bits, entonces, su capacidad es de 2^8=256 direcciones de memoria.

La siguiente figura representa un diagrama en bloques de una memoria.

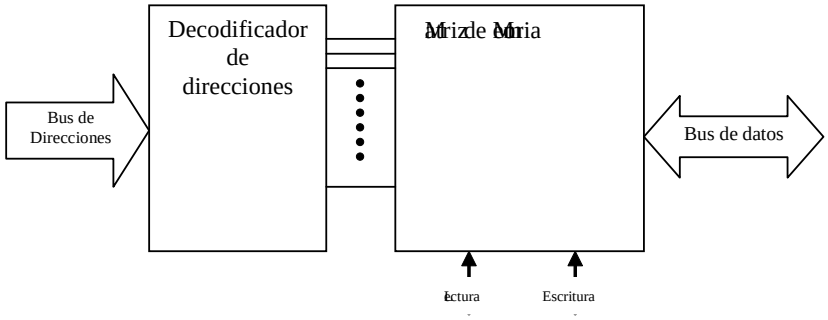


Figura 8.1 Diagrama en bloques de una memoria

8.1.1 Operación de escritura

En la operación de escritura de una memoria, se introduce en el bus de direcciones un código que corresponde a la dirección en la que se desea almacenar el dato. El decodificador de direcciones decodifica el valor colocado en el bus de direcciones seleccionando el registro en el cual se almacena el dato. Posteriormente, se debe colocar en el bus de datos, el dato que se desea almacenar. Finalmente se debe dar una orden de escritura para que el dato quede almacenado en la memoria. Cuando se escribe un nuevo dato en una posición de memoria, el dato anterior es sobreescrito, lo cual indica que ese dato anterior se destruye.

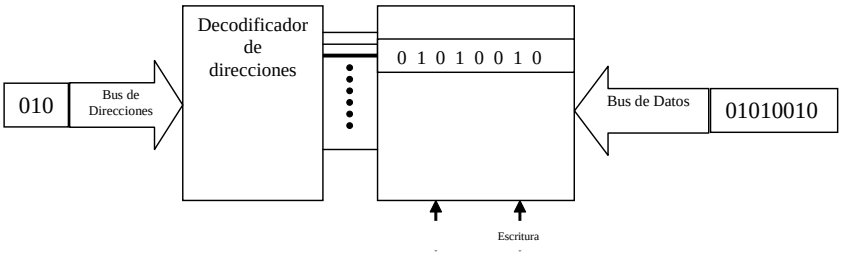


Figura 8.2 Ilustración de operación de escritura

8.1.2 Operación de lectura

En la operación de lectura de una memoria, se introduce en el bus de direcciones un código que corresponde a la dirección de la que se desea capturar o leer el dato. El decodificador de direcciones decodifica el valor colocado en el bus de direcciones seleccionando el registro del cual se toma el dato. Finalmente, se debe dar una orden de lectura para que el dato se coloque en el bus de datos. Una vez esté en el bus de datos, se puede hacer uso de dicha información. Cuando se lee un dato de una posición de memoria, el dato se mantiene almacenado. Esto permite darle nuevo uso a dicho dato.

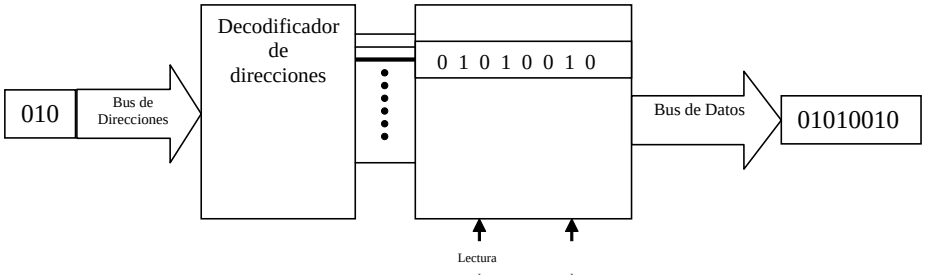


Figura 8.3 Ilustración de operación de lectura

8.2 MEMORIAS DE SÓLO LECTURA ROM

ROM son las siglas de *read-only memory*, que significa «memoria de sólo lectura». Las memorias ROM mantienen de forma permanente los datos almacenados, los cuales, pueden ser leídos pero no modificados (memoria no volátil). Una memoria ROM almacena datos que se requieren constantemente en las aplicaciones. La ROM suele almacenar la configuración del sistema o el programa de arranque de un computador.

Las memorias de sólo lectura o ROM son utilizadas como medio de almacenamiento de datos en los computadores, debido a que no se puede escribir fácilmente, su uso principal reside en la distribución de programas que están estrechamente ligados al soporte físico del computador, y que seguramente no necesitarán actualización. Por ejemplo, una tarjeta gráfica puede realizar algunas funciones básicas a través de los programas contenidos en la ROM.

Una razón de que se utilice la memoria ROM para almacenar datos es la velocidad ya que los discos son más lentos. Además no se puede leer un programa que es necesario para ejecutar un disco desde el propio disco. Por lo tanto, la BIOS, o el sistema de arranque oportuno de la computadora normalmente se encuentran en una memoria ROM.

Las ROMs semiconductoras se fabrican con tecnología bipolar o con tecnología MOS (Metal-Óxido Semiconductor). La figura 8.4 muestra la clasificación de las memorias ROM.

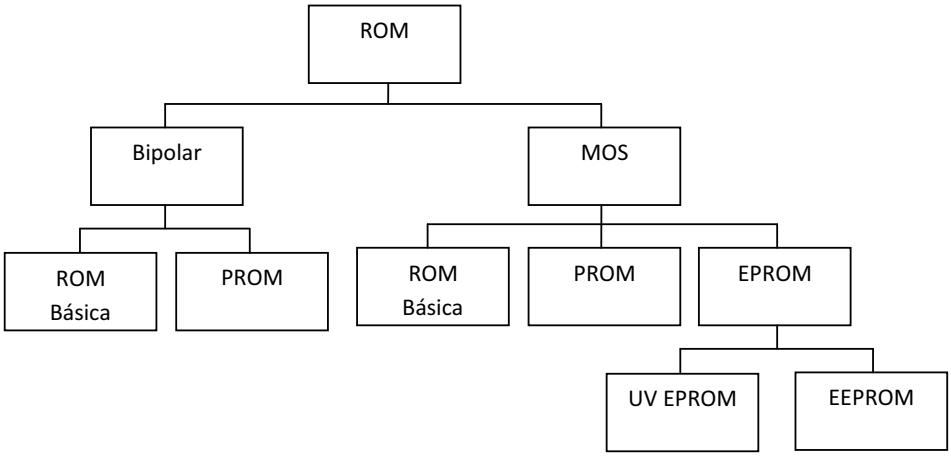


Figura 8.4 Tipos de memorias ROM

8.2.1 ROM Básica

La ROM básica es una memoria programada permanentemente durante el proceso de fabricación para proporcionar funciones estándar. La mayoría de los circuitos integrados ROM utilizan la presencia o ausencia de una conexión de transistor en una unión fila-columna para representar un 1 o 0. Las celdas de una memoria pueden ser bipolares, es decir, que están construidas a base de transistores BJT (*Bipolar Junction Transistor*) o pueden ser construidas a base de transistores MOSFET (*Metal-Oxide Semiconductor Field Efect Transistor*).

La representación de una memoria ROM pequeña, por ejemplo, una memoria de 256X4, que indica que tiene 256 posiciones de memoria y cada posición es de 4 bits, se realiza mediante una matriz de celdas de 256 filas por 4 columnas. Esta memoria tiene entonces 8 bits en el bus de direcciones y 4 bits en el bus de datos.

8.2.2 PROM

Las PROM (*ROM Programable*) se encuentran disponibles en tecnología bipolar y en tecnología MOS. Generalmente estas memorias tienen buses de datos de 4 u 8 bits y capacidades de alrededor de 250.000 posiciones de memoria. Las PROM utilizan mecanismos de fundición para almacenar bits donde una

celda se funde o no para representar un 1 o 0 lógico, entonces una memoria PROM utiliza fusibles en cada celda. El proceso de la fundición es irreversible. Una vez programada no se puede cambiar.

Durante el proceso de programación de la memoria, a cada celda se le aplica una corriente suficiente para fundir el fusible de la celda. De esta forma esta celda queda abierta y representaría un 0 lógico.

Normalmente una memoria PROM se programa colocándola en un dispositivo denominado programador de PROM. El proceso de programación consiste en colocar una dirección en el bus de direcciones y enviar un impulso que produzca corriente a cada línea del bus de datos para fundir los fusibles requeridos. De esta forma queda almacenado el dato.

8.2.3 EPROM

Una EPROM es una PROM borrable. Una EPROM puede ser reprogramada, si antes se borra los datos que contenga la memoria. Utiliza matrices NMOS-FET con estructura de puerta aislada. La puerta del transistor aislada no tiene conexión eléctrica y puede almacenar una carga eléctrica durante un período de tiempo indefinido. Los bits en esta matriz se representan mediante la presencia o ausencia de una carga almacenada en la puerta. El borrado consiste en la eliminación de la carga de la puerta. Los tipos fundamentales de EPROM son las UVPROM (*PROM Borrable* por Ultravioleta) y las EEPROM (PROM Eléctricamente Borrable).

8.2.4 UVPROM

En una UVPROM la puerta aislada del FET está flotando dentro de un material óxido aislante. El proceso de programación hace que los electrones sean eliminados de la puerta flotante. El borrado se realiza mediante la exposición del chip de la matriz de memoria a una radiación ultravioleta de alta intensidad a través de una ventana de cuarzo que posee la memoria en la parte superior del encapsulado. La carga positiva almacenada en la puerta se neutraliza después de un período de tiempo de exposición de la luz ultravioleta sobre la memoria.

8.2.5 EEPROM

Las PROM Borrables eléctricamente, se pueden borrar y programar mediante impulsos eléctricos. Las EEPROM se pueden reprogramar dentro del propio circuito, lo que permite reconfigurar cualquier sistema. Los dos tipos de EEPROM son los MOS de puerta flotante y la de silicio de óxido nitroso (NMOS, *Metal Oxide Silicon*). La aplicación de una tensión en la puerta de control de la

estructura de puerta flotante permite la eliminación y el almacenamiento de la carga en la puerta flotante.

8.3 MEMORIAS DE ACCESO ALEATORIO RAM

En una memoria RAM los datos se pueden leer y escribir de forma aleatoria, es decir, que se puede leer y escribir de una posición de memoria deseada. Además, los tiempos de respuesta en una memoria RAM son bajos, esto indica que es una memoria supremamente eficiente. Al escribir un dato en una posición de la memoria, los datos anteriores son reemplazados por el nuevo dato. Sin embargo, cuando los datos son leídos de una posición de memoria, estos datos se conservan. La memoria RAM puede almacenar información, mientras tenga energía eléctrica.

Las memorias RAM, también se fabrican con tecnologías bipolar y MOS. Hay otras memorias que usan ambas tecnologías denominadas BiMOS. La figura 8.5 muestra la clasificación de memorias RAM.

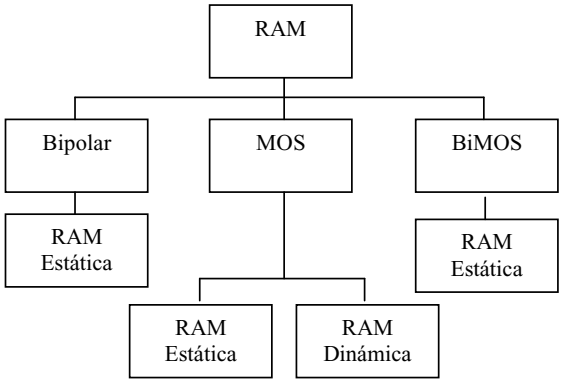


Figura 8.5 Tipos de memorias RAM

8.3.1 Arquitectura de RAM Estática (SRAM)

Una memoria SRAM se direcciona del mismo modo que una memoria ROM. La principal diferencia entre la organización de las SRAM y las PROM es que las SRAM, tienen bus de datos bidireccional, es decir de entrada y salida y tienen unidad de control de lectura y escritura de información.

En modo de lectura, se coloca en el bus de datos el dato almacenado en la posición de memoria correspondiente a la dirección colocada en el bus de direcciones. En modo de escritura, el dato colocado en el bus de datos, se al-

macena en la posición de memoria correspondiente a la dirección colocada en el bus de direcciones. El bus de datos se compone de las líneas llamadas DQ_0 a DQ_7. En el proceso de escritura, estas líneas se denominan D_0 a D_7 y en el proceso de lectura estas líneas se denominan Q_0 a Q_7.

En la figura 8.5 se muestra la organización de una memoria SRAM. La matriz de celdas de memoria está organizada en 256 filas y 128 X 8 columnas, es decir, 128 columnas cada una de 8 bits. El modo de operación de la SRAM es el siguiente: la entrada de habilitación E, debe estar a nivel bajo para que el circuito se encuentre habilitado, 8 de las 15 líneas de dirección se decodifican en el decodificador de filas. Esto indica que se selecciona solo una de 256 filas. Las líneas restantes de dirección las decodifica el decodificador de columnas. Esto indica que se selecciona solo una de 128 columnas de 8 bits.

En el proceso de lectura, la entrada de habilitación W debe estar a nivel alto y la entrada de habilitación G, debe estar activa a nivel bajo. Las compuertas negativa AND, posterior a las líneas de habilitación, permiten activar los buffer triestado del bus de datos, de tal forma que los datos almacenados, se presenten en el bus de datos.

En el proceso de escritura, la entrada de habilitación W debe estar activa a nivel bajo y la entrada de habilitación G, debe estar a nivel alto. Las compuertas negativa AND, posterior a las líneas de habilitación, permiten activar los buffer triestado del bus de datos, de tal forma que los datos colocados en el bus de datos, ingresen a la matriz de memoria en la fila y columna seleccionada.

8.4 EXPANSIÓN DE MEMORIAS

Las memorias disponibles se pueden ampliar para incrementar la longitud de palabra, es decir, el número de bits de los datos o para ampliar la cantidad de posiciones de memoria, es decir, ampliar el número de direcciones de la memoria.

8.4.1 Expansión de longitud de palabra

El objetivo de la expansión de la longitud de la palabra es ampliar el bus de datos. Para ello, es recomendable utilizar dos memorias RAM o ROM idénticas.

La figura 8.6 muestra el símbolo de una memoria ROM de 32 X 8.

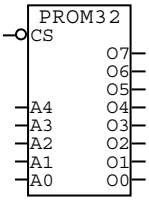


Figura 8.6 Memoria PROM 32 X 8

Con base en la memoria anterior, se puede ampliar el número de bits del bus de datos al doble, utilizando dos circuitos integrados de esta memoria. En este caso se obtendría una memoria PROM de 32 X 16. La implementación de esta memoria expandida sería la siguiente (véase figura 8.7):

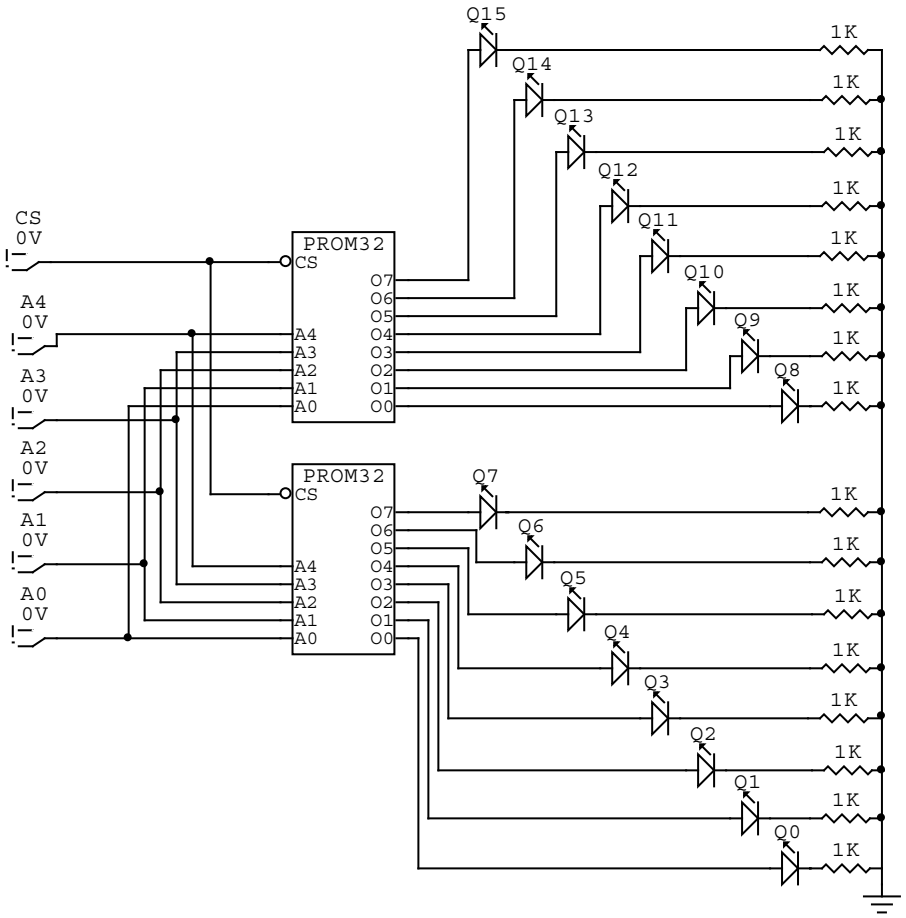


Figura 8.7 Expansión de longitud de palabra en memoria PROM a 32 X 16

En la expansión anterior, se debe colocar el bus de direcciones en común para ambas memorias y la salida se amplía al doble de cada memoria.

De igual forma se implementa para una memoria RAM. La figura 8.8 muestra el símbolo lógico de una memoria RAM de 1Kb X 8.

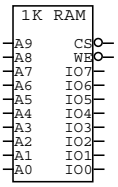


Figura 8.8 Memoria RAM 1Kb X 8

Con base en esta memoria de 1Kb X 8, la implementación de la memoria expandida sería la siguiente (véase figura 8.9):

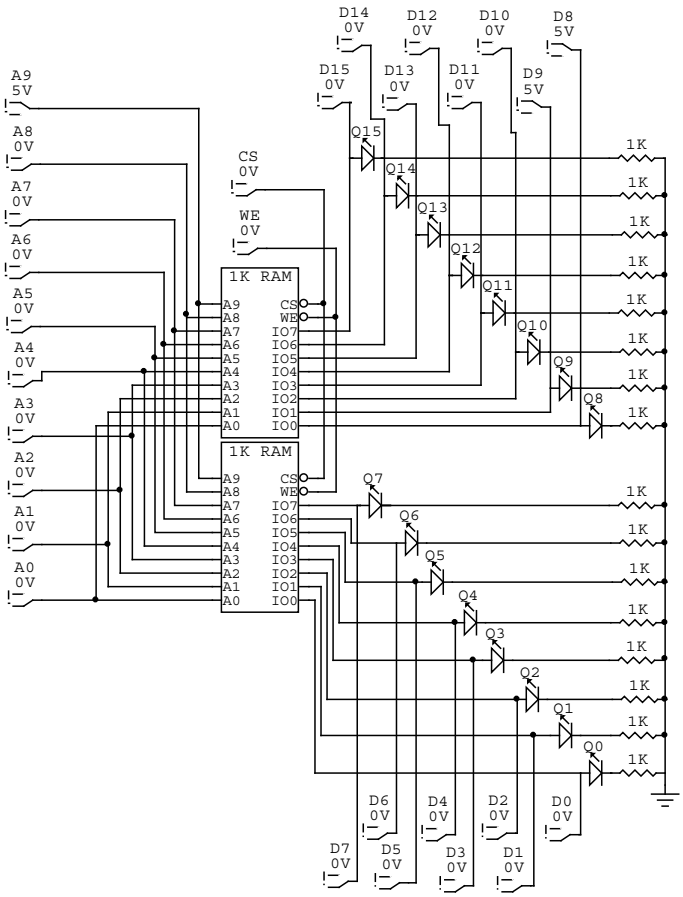


Figura 8.9 Expansión de longitud de palabra en memoria RAM a 1Kb X 16

En la implementación anterior, las entradas del bus de direcciones van en común y el bus de datos expande. Sin embargo, es necesario tener en cuenta que el bus de datos es bidireccional. En esta memoria, al colocar la línea WE en 0, la memoria se encuentra en estado de escritura y al colocar la línea WE en 1, la memoria se encuentra en estado de lectura. Esto indica que el bus de datos es de entrada cuando WE es 0 y de salida cuando WE es 1. Por tal motivo, en este ejemplo se ha conectado diodos LED para visualizar la salida y *dipswitch* para proporcionar datos de entrada.

8.4.2 Expansión de capacidad de almacenamiento o tamaño

El objetivo de la expansión de la capacidad de almacenamiento es ampliar el bus de direcciones en un bit. Al hacer esta operación, se expande la capacidad de la memoria al doble.

Con base en la memoria PROM 32 X 8, se puede ampliar la capacidad de la memoria al doble usando dos memorias idénticas, resultando una memoria expandida de 64 X 8. La implementación de esta memoria expandida sería la siguiente (véase figura 8.10):

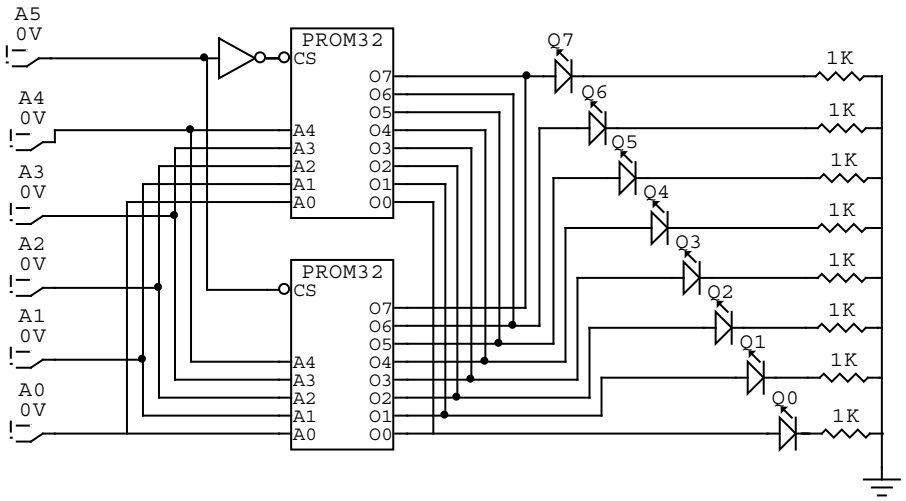


Figura 8.10 Expansión de tamaño en memoria PROM a 64 X 8

En la expansión anterior, se debe colocar el bus de direcciones en común para ambas memorias. Además la entrada CS, se convierte en el bit de mayor peso del bus de direcciones, de tal forma que cuando es 0 habilita la memoria inferior y cuando es 1 habilita la memoria superior. Las salidas se conectan en común.

Esta expansión con memoria RAM también es posible. Con base en la memoria RAM de 1Kb X 8, la implementación de la memoria expandida sería la siguiente (véase figura 8.11):

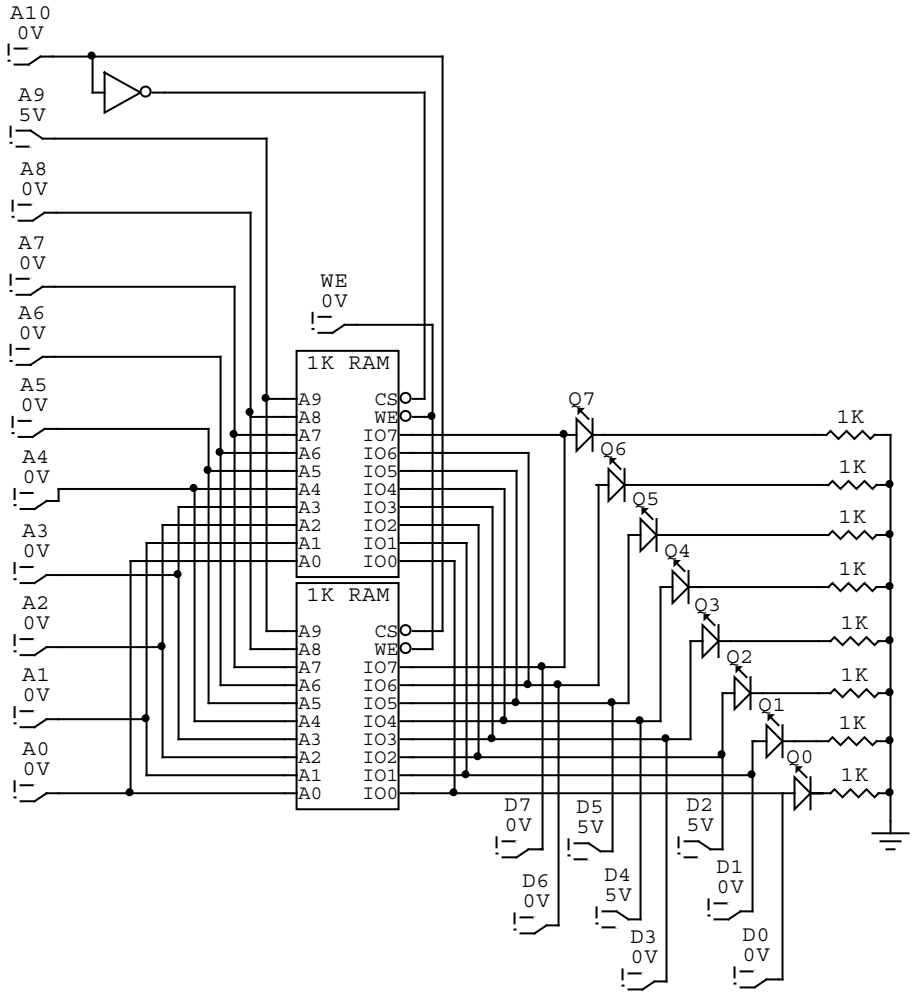


Figura 8.11 Expansión de tamaño en memoria RAM a 2Kb X 8

En la expansión anterior de la figura 8.11, se debe colocar el bus de direcciones en común para ambas memorias. Además la entrada CS, se convierte en el bit de mayor peso del bus de direcciones, de tal forma que cuando es 0 habilita la memoria inferior y cuando es 1 habilita la memoria superior. Además, en esta memoria, al colocar la línea WE en 0, la memoria se encuentra en estado de escritura y al colocar la línea WE en 1, la memoria se encuentra en estado de lectura. Esto indica que el bus de datos es de entrada cuando WE es 0 y de salida cuando WE es 1. Por tal motivo, en este ejemplo se ha conectado diodos LED para visualizar la salida y dipswitch para proporcionar datos de entrada.

Ejercicios Propuestos

1. Diseñar e implementar una memoria ROM con capacidad de 1K x 8, basado en memorias ROM de 512 x 8.

2. Diseñar e implementar una memoria ROM con capacidad de 1K x 8, basado en memorias ROM de 1K x 4.

3. Diseñar e implementar una memoria ROM con capacidad de 1K x 8, basado en memorias ROM de 512 x 4.

4. Diseñar e implementar una memoria RAM con capacidad de 1K x 8, basado en memorias ROM de 512 x 8.

5. Diseñar e implementar una memoria RAM con capacidad de 1K x 8, basado en memorias ROM de 1K x 4.

6. Diseñar e implementar una memoria RAM con capacidad de 1K x 8, basado en memorias ROM de 512 x 4.

BIBLIOGRAFÍA

ECG Semiconductors (1994). *Master Replacement Guide*. Philips. 16th edition.

ECG Semiconductors (1995). *TTL Digital Integrated Circuits Data Manual*. Philips.

FLOYD, Thomas (2005). *Fundamentos de Sistemas Digitales*. Pearson. Novena edición.

MANO, Morris (1994). *Lógica Digital y Diseño de Computadores*. Prentice Hall. Tercera edición.

TOCCI, Ronald, WIDMER Neal (2003). *Sistemas Digitales, Principios y Aplicaciones.* Pearson. Octava edición.

TOKHEIM, Roger (1995). *Principios Digitales*. McGraw-Hill. Tercera edición.

VYEMURA, John (2000). *Diseño de Sistemas Digitales, Un enfoque integrado*. Thomson.

INFOGRAFÍA

Video Introducción a Circuit Maker. Tomado de http://www.hectorflorez.com/web/videosexposiciones/IntroduccionCircuitMaker/. Fecha: 30 de mayo de 2010.

Datasheet Catalog. Tomado de http://www.datasheetcatalog.com/. Fecha: 30 de mayo de 2010.

Display de siete segmentos. Tomado de http://www.cypsela.es/especiales/pdf195/display.pdf. Fecha: 30 de mayo de 2010.

Visualizador de siete segmentos. Tomado de http://es.wikipedia.org/wiki/Visualizador_de_siete_segmentos. Fecha: 30 de mayo de 2010.

Circuitos digitales. Tomado de http://www.monografias.com/trabajos14/circuidigital/circuidigital.shtml. Fecha: 30 de mayo de 2010.

Circuitos Integrados. Tomado de http://www.profesormolina.com.ar/electronica/componentes/int/index.htm. Fecha: 30 de mayo de 2010.

En esta obra encontrará el material necesario para abordar el tema de circuitos lógicos combinacionales y secuenciales como fundamento de la electrónica digital, la cual contiene diferentes dispositivos que conllevan a comprender la construcción de la arquitectura del computador.

A través de ocho capítulos se describen conceptos de sistemas numéricos, compuertas lógicas, álgebra de Boole, lógica combinacional, elementos básicos de almacenamiento, contadores, registros y memorias.

- Conceptos básicos
- Ejemplos
- Problemas propuestos
- Glosario técnico

Héctor Arturo **Flórez** Fernández

Ingeniero Electrónico e Ingeniero de Sistemas de la Universidad El Bosque, Magíster en Ciencias de la Información y las Comunicaciones de la Universidad Distrital Francisco José de Caldas, Especialista en Alta Gerencia y candidato a Magíster en Gestiones de Organizaciones de la Universidad Militar Nueva Granada.

Certificación CCNA Discovery (Cisco Certified Network Asociated). Docente de la Fundación Universitaria Konrad Lorenz y de la Universidad Distrital Francisco José de Caldas. Par Académico del Ministerio de Educación, distinguido con Excelencia Académica por la Universidad Distrital Francisco José de Caldas, Profesor Distinguido en la Ciencia por la Fundación Universitaria Konrad Lorenz y Mención Grado de Honor en Ingeniería Electrónica por la Universidad El Bosque.

Autor de los artículos: *Inteligencia artificial mediante ingeniería de software; Construcción de Ontologías OWL; y La web semántica y sus posibles aplicaciones en el ámbito universitario*, publicados en revistas indexadas de reconocimiento internacional.

de la ediciones U
conocimiento a su alcance
www.edicionesdelau.com

E-learning en:
www.aprendizajeenlinea.com
www.edicionesdelau.com

ISBN 978-958-994-900-9